マイコンとパズルの世界

BASICプログラミング頭の体操

小谷善行 著

産報出版

①問題を分析し、
②問題を解く手順を設計し、
③その手順をプログラムとして実現し、
④データを与えてプログラムを動かし答を得る。
ということを行なって初めて「問題をパソコンで解いた」ことになります。(中略)
パズルをパソコンに解かせることができるようになった人は、単に楽しいだけでなく、
知らないうちにソフトウエアの作成能力を身につけているに違いありません。
この能力を育て、仕事、趣味、勉強に活用していただければ理想的です。
〈「まえがき」より〉

イラスト　桜井　一　　装丁　白土健康

マイコンとパズルの世界

BASICプログラミング頭の体操

小谷善行 著

産報出版

マイコンとパズルの世界

BASICプログラミング頭の体操

小谷善行 著

産報出版

まえがき

人間が頭をひねって考えても，なかなかできない難問のパズルをマイコン（パソコン）が簡単に解いてしまう。こういうことを自分の手でさせられれば楽しいと思いませんか。それがもう，だれにでもできる時代になったのです。パズルだけでなく，言葉による対話，将棋や麻雀などのゲーム，定理の証明など，人間だけができると思われていた複雑な情報処理も，少しずつ計算機の上で実現しつつあります。このようなことを自分でマイコンの上でやってみるのも，ひまさえあれば十分可能性のあるものになってきました。

計算機は電卓から超大型のものまで，機能が同じものを比べれば，年率25～30％で価格が下がっています。パソコンも最も基本的なシステムで十数万円から十万円を切る方向に進んでいます。カラーテレビを買い換える代わりにパソコンを買う，というようなことをすれば，個人でも簡単にパソコンを持つことができます。つまり難問を解くための道具は，すぐ手にはいるわけです。

あとはその道具の**使い方**，ここがキーポイントです。つまり，

(1) 問題を分析し，

(2) 問題を解く手順を設計し，

(3) その手順をプログラムとして実現し，

(4) データを与えてプログラムを動かし答を得る。

ということを行なって初めて「問題をパソコンで解いた」ことになります。しかも，このことは人間がしなければなりません。テレビは技能がなくてもスイッチは入れられますが，自動車は練習しなければ動かせません。この点でパソコンも自動車の場合と同じです。(1)から(4)までを行なわなければ，答は出てきません。

この本は「こうやれば問題を解けるのだ」というところを実地に示していきます。ですから，この本はBASICのプログラミングの本とい

うよりもむしろ，パズルを題材にした「問題を解く方法を考える本」といえます。

この本を通じて，読者に難問のパズルをパソコンに解かせることができるようになっていただければ，うれしく思います。パズルをパソコンに解かせることができるようになった人は，単に楽しいだけでなく，知らないうちにソフトウエアの作成能力を身につけているに違いありません。この能力を育て，仕事，趣味，勉強に活用していただければ理想的です。

この本で紹介したプログラムは，全部著者が自分の手で（楽しみながら）設計，作成したオリジナルなものです。プログラムをキーインして動かしてみるだけでは面白くありません。このプログラムを参考にして，別のパズルのプログラムを自作することに挑戦してみてください。うまくいったら著者にも教えてください。

この本を書くにあたっては，東京農工大学数理情報工学科の方々，パズル懇話会の方々との間の雑談や議論から得られたいろいろな知識が役に立っています。また産報出版の木内雄一氏には企画から刊行まで，たいへんお世話になったことを感謝します。

昭和56年10月

小谷善行

もくじ

0 マイコンとパズルの世界の始まり … 7

1 論理パズル1　仲良し仲間が海へ行く ………… 15

2 論理パズル2　本を並べたりお見合いさせたり… 25

3 虫食算を解く ………………………………… 35

4 覆面算を解く ………………………………… 53

5 切符の数字　数楽パズル ………………… 65

6 幸福の手紙　自己を表示するプログラム……… 75

7 桂馬道の問題 ………………………………… 85

8 碁石パズル　「ひろいもの」 ……………… 105

9 迷路パズル　囲碁の表示システムを作る……… 119

10 詰碁を解く ………………………………… 139

11 麻雀のパズル ……………………………… 157

12 しゃべる機械 ……………………………… 169

13 「課題」の解答 …………………………… 182

14 「休憩室」の解答 ………………………… 208

休憩室

1　この問題はコの問題　*24*

2　変数名はよく考えて付けよう　*34*

3　0は1を生み，1は2を生み，2は3を生み，…　*52*

4　FORループを0回まわる　*64*

5　1980年（昭和55年）は何の年　*74*

6　まっすぐ進めない蟻　*84*

7　ナイトは将棋盤がきらい？　*104*

8　100行のプログラム10個と1000行のプログラムを1個ではどちらが作りやすい？　*117*

9　1キロは1000ですか？　*138*

Tの話：コンパイラとインタープリタ……… *81*

模様を描く……………………………………… *211*

参考になる本……………………………… *213*

本書で説明しているソフトウエア関連用語集…… *6*

まえがき…………………………………… *2*

装丁　白土健康

カバー・扉イラスト　桜井　一

本文イラスト　岡安孝男

本書で説明しているソフトウエア関連用語集

すべての組合せ…………………………… 17

字下げ………………………………………… 20

順列…………………………………………… 28

わかりやすいプログラムを書こう……… 41

組合せ論的爆発…………………………… 47

サブルーチンの役割……………………… 60

引数…………………………………………… 61

２分木………………………………………… 68

数の精度と誤差…………………………… 70

コンパイラとインタープリタ…………… 81

なるべくデータで表わす………………… 89

スタック……………………………………… 90

再帰的プログラム………………………… 96

縦型探索…………………………………… 98

モジュール化……………………………… 114

共通の機能をモジュールにする………… 129

後もどり…………………………………… 131

人間とのインターフェース……………… 131

ANDとOR ……………………………… 141

ゲーム木…………………………………… 142

枝刈り……………………………………… 143

人工知能…………………………………… 170

学習………………………………………… 174

ハッシュ法………………………………… 177

0★マイコンとパズルの世界の始まり

一家に1台
＝個人用計算機（パソコン）の時代がやって来た

最近のマイコン（パソコン）やBASIC電卓の普及には目をみはるものがあります。だれでもマイコンを所有でき，自由に使える時代が来ました。マイコンのソフトウエア（プログラム）も非常にたくさんのものが作られています。なかでもホビーの分野をながめてみると，インベーダーを初めとする実にさまざまなテレビゲームが作られ，マイコン雑誌にのり，カセットにして売られています。しかし，パズルもマイコンもブームであるのに，**パズルを解くプログラム**はあまり多く見られません。

頭脳労働をマイコンにもやらせよ
＝マイコンでパズルを解いてみよう

マイコンはすばらしい能力を持った小さな巨人です。人間がぜったいできないような複雑なパズルも簡単に解いてしまいます。また，マイコンは可能性をひとつ残らず全部調べるというのも得意です。場合によっては新しいパズルを作り出すこともできます。このようなマイコンの大きな能力をひき出すのが，その中に入れるソフトウエア（プログラム）なのです。しかし，パズルを解くプログラムは，まだあまり多く作られていないし，作り方もよく知られているわけではありません。「この点を書いた本はなぜないのだろう？　ないなら自分で書こう」というのが，この本を作ることになった一つの理由です。ですからこの本の主題は**パズルを解くプログラムを作る**ということです。プログラミング言語は色々なマイコンにもっとも普及しているBASICを使います。人工知能の分野ではLISPなどの言語が使われ，パズルのプログラムも作られていますが，一般にはなじみがありません。この本では皆が知っているBASICを使ってパズルを解くプログラムの楽しさをお知らせしたいと思います。

作るアホウに使うアホウ，同じアホウなら……
＝プログラムを自分で作るのが大事

パソコンを買っても，ゲームのプログラムのカセットを買ってきたり，プログラムをコピーして遊ぶだけでは，しばらくするとあきてくるし，発展がありません。それではどうすればいいか？ **プログラムを自分で書く**ことです。パズルのようなある程度複雑なプログラムを作ることです。むずかしそうに見えても，着実に考えればだんだんわかってきます。マイコンはプログラムを作った苦労に必ず応えてくれるはずです。マイコンがむずかしいパズルの答を出してきたとき，あるいはまるで人間がするような反応をしてきたときなど，ほんとうにうれしいものです。

“大地主と大貧民”の差？
＝未来ではプログラムを書けると書けないは大違い

プログラムが自由に書けるようになるということは，趣味の楽しさがふえるだけでなく，これからの世の中では，いろいろな得をもたらすでしょう。10年もすれば職場や家庭にパソコンが，今の電卓のようにゴロゴロしているようにきっとなるにちがいありません。そのときにはプログラムを書けるという能力は，自動車を運転できるという能力以上に大事なものとなるでしょう。「将来人間は２種類に分かれる。一方は計算機と対話（プログラミング）できる人間，他方はそれができない人間である」という人もいます。この考えは極端ですが，自動車が運転できるかできないかということ以上の差は生まれてくるでしょう。つまり車が運転できるようになれば，普通その人の行動範囲は広くなります。それと同じあるいはそれ以上に計算機と対話できるということは,人の行動範囲を広げるに違いありません。しかも自動車の運転のように動きまわる能力を高めるのではなく，人間の情報処理（考える，計算する，覚えておく，決める，整理する）の能力を高めるものなのです。この本を通じて，みなさんにはぜひ自由にプログラムを書けるようになってもらいたいと思います。

習うより慣れよ!!
＝プログラミングの入門書ではわからないこと

この本はプログラミングの入門書ではありません。この本を書くにあたって感じたことは，プログラミングの入門書は多いが「それから先」をどうすればよいかを書いた本が少ないということです。この本の目ざすのは「パズルを解くプログラムを作る」ということです。この目的に進む途中で自然と「それから先」つまりプログラミングの技法や情報工学の考え方がわかるようにしたいと思います。このように実際的に物を作ったりするうちに勉強できてしまうのをOJΓ（on the job training）といいます。何年間も学校で習った英語よりも数か月アメリカやイギリスに行って毎日使った英語の方がものになっているという話があります。プログラミングも同じです。パズルのプログラムを作っているうちに自然にプログラミングの能力が身についてしまう。スゴイと思いませんか？（でもこういうことは考えずにパズルを楽しむ方がいいかもしれないけれど）。

「それから先」＝入門の次に来るもの

では「それから先」の中身はなんでしょうか？。ソフトウエアで使われている基本的な仕組みがあります。スタック，再帰的サブルーチンなどがそうです。計算機科学や人工知能の基本的な概念があります。組合せ論的爆発，縦型探索などがそうです。よいプログラムの書き方に関しても字下げや変数名の名付け方などいろいろなものがでてきます。ソフトウエアの設計法も少し触れます。このようないろいろな知識を活用するとよいソフトウエアが作れるのです。力まかせにプログラムを書いていてもなかなかよいプログラムはできません。でも，むずかしそうなことが書いてあっても気にしないでください。わからないところは気が向いたらパズルのプログラムをいろいろ作った後で読み返すのでよいのです。

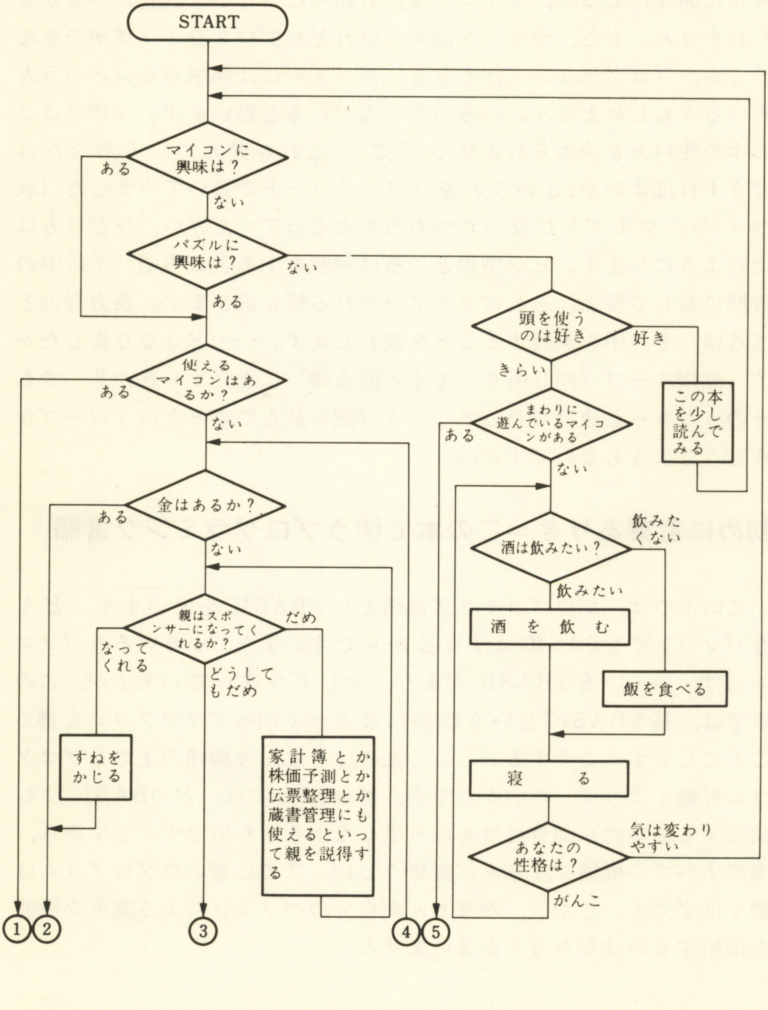
読者のためのフローチャート＝この本の使い方

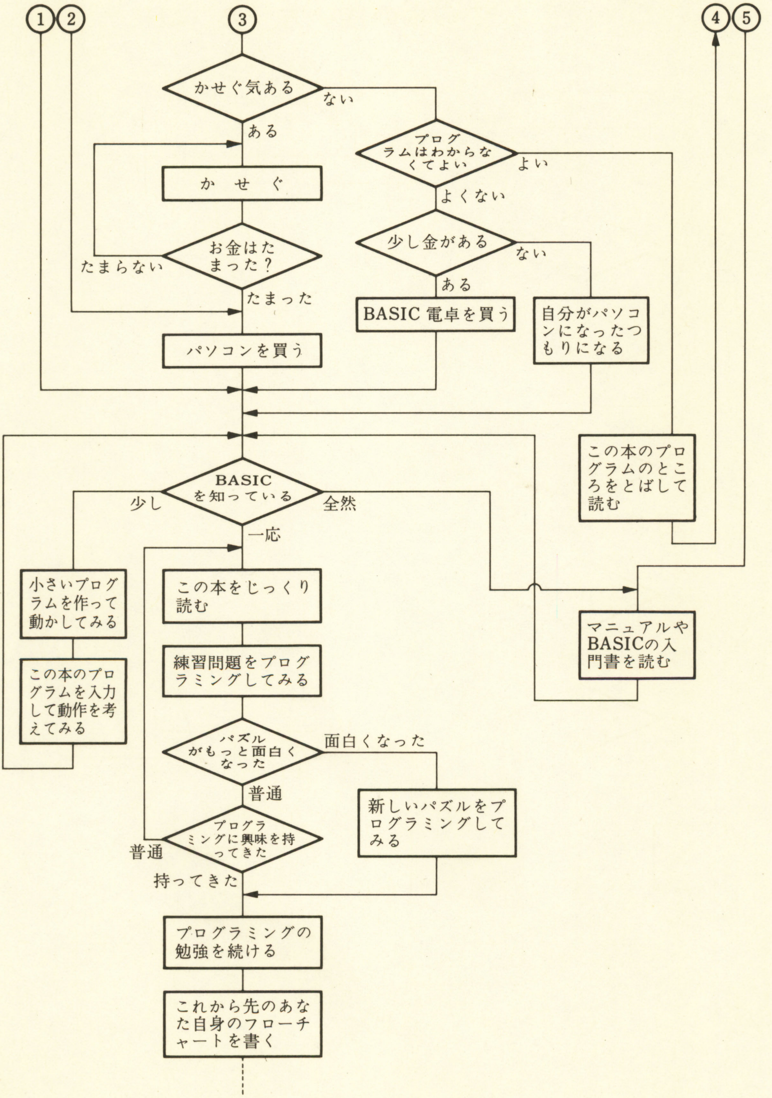
この本はプログラミングの入門書ではない，といいました。でもパズルに興味はあるが，マイコンは何も知らないという読者がいるかもしれません。また，マイコンはあるけれどもプログラミングができないとか，プログラミングはできないがパズルには興味がないという人がいるかもしれません。いろいろな人がいると思います。一概にはこの本の使い方を決められません。そこで「これを読んでいるあなたはどうすればよいか」というのをフローチャートで書いてみました（次ページ）。マイコンになったつもりでたどってください。たどり方は次のようにします。ヒシ型のところは判断をするところで，その中の質問に対して答え，それによって分かれる枝を選びます。長方形のところは，その中を実行することを表わします。……どうなりましたか？　無限ループ（同じ所をぐるぐる回る輪）になりましたか？　でもそのうちきっと抜けられるでしょう（酒を飲んで寝るというループならどうしようもないですが）。

初めに言葉ありき＝この本で使うプログラミング言語

この本では，プログラミング言語としてBASICを使います。どんなパソコンでもBASICは使えるからです。また，いろいろなパソコンには，いろいろなBASICがあり，少しずつ違っていますが，この本では，基本BASICという規格になるべく沿ってプログラムを書くことにします。こうすることにしたのは，どんな機種の上でもプログラムが動くようにしたいからです。基本BASICは，どのBASICにも共通する基本的な部分だけを取り出したようなものです。ですから，実際すべての機種でほとんど変更なしに，ここに書いたプログラムは動くはずです。しかし，みなさんが自分のパソコンにある高度な機能を活用するのはもちろんかまいません。

「あなたはどうすればよいか」と「この本の使い方」のフローチャート




①
②
③
④
⑤
かせぐ気ある
ない
ある
プログラムはわからなくてよい
よい
よくない
かせぐ
少し金がある
ない
ある
お金はたまった？
たまらない
たまった
BASIC電卓を買う
自分がパソコンになったつもりになる
パソコンを買う
この本のプログラムのところをとばして読む
BASICを知っている
少し
全然
一応
小さいプログラムを作って動かしてみる
この本をじっくり読む
マニュアルやBASICの入門書を読む
練習問題をプログラミングしてみる
この本のプログラムを入力して動作を考えてみる
パズルがもっと面白くなった
面白くなった
普通
新しいパズルをプログラミングしてみる
プログラミングに興味を持ってきた
普通
持ってきた
プログラミングの勉強を続ける
これから先のあなた自身のフローチャートを書く

1★論理パズル1

仲良し仲間が海へ行く

【問　題】

６人のクラスメートのあいだで海へ行く話が持ち上りました。しかし，いろいろな事情があってみんなが行けるわけではありません。最後に何人かが海に行くことになりましたが，どんな組合せだったでしょうか？

６人の名前はアヤコさん，イクコさん，ウーちゃん，エイコさん，オウイチ君，カズオ君です。６人のうちで最低３人は集まらないとつまらないので海へ行きません。

アヤコさんとカズオ君はけんかをしているので，アヤコさんは，カズオ君といっしょには行かないし，カズオ君が行かないなら私は行く，といっています。カズオ君も，アヤコさんとはいっしょに行かないし，アヤコさんが行かないなら行くといっています。

イクコさんとエイコさんは仲良しで，いつもいっしょに行動しています。

オウイチ君はウーちゃんが行くなら行くといっています。

ウーちゃんとカズオ君は仲が悪いので，２人はいっしょに海には行かないといっていますが，２人ともイクコさんが好きなので，イクコさんが海へ行くなら２人はいっしょに海へ行くかもしれません。

アヤコさん，イクコさん，ウーちゃん，エイコさんは交代でアルバイトをしています。そのため，このうち１人は残って働かなければなりません。海へは車で行くことになっています。車を持っているのはアヤコさんとオウイチ君だけなので，このうち１人は海へ行くことになります。

女だけで行くと，痴漢が出てきたりしたら困るので，ウーちゃん，オウイチ君，カズオ君のうち１人は海に行く必要があります。

アヤコさんは自分１人だけ残されて，あとの５人が海へ行ってしまうのは絶対いやだといっています。

以上が６人の“事情”です。

さてこの問題を頭で考えてできますか？　パズルの本などに出てくる問題よりも少しむずかしくしてみました。BASICのプログラムを作って解けば簡単であることをわかってもらいたかったからです。でも少し時間をかければ，頭で考えてもできない問題ではありません。

●すべての可能性を考える

このような論理パズルを解くときには，「とり得る全部の場合とはどんなものか」を考えることが大切です。今６人のうち何人かが海へ行くということを考えているわけです。１人ひとりについては「海に行く」と「海に行かない」の２通りの可能性があります。したがって，６人では，

$$2\times2\times2\times2\times2\times2=2^6=64\text{通り}$$

の可能性があるわけです。この64通りのなかでいろいろな都合が全部いいかどうか調べればよいのです。この64通りを作り出すのは次のようにすれば簡単です。アヤコさんが海へ行くかどうかを表わす変数をAとします。Aが１のときアヤコさんは海へ行くことを，Aが０のときアヤコさんは海へ行かないことを表わすものとします。同じようにして，イクコさん，ウーちゃん，エイコさん，オウイチ君，カズオ君が海へ行くかどうか表わす変数をそれぞれ，I，U，E，O，Kとします。こうすると，

```
FOR A=0 TO 1
 FOR I=0 TO 1
  FOR U=0 TO 1
   FOR E=0 TO 1
    FOR O=0 TO 1
     FOR K=0 TO 1
      ┌──────────┐
      │一つの場合の調査│
      └──────────┘
     NEXT K
    NEXT O
   NEXT E
```

```
        NEXT U
      NEXT I
    NEXT A
```

とすることによって全部の組合せを実現できます。「1つの場合の調査」のところに,

```
PRINT A; I; U; E; O; K
```

を入れると

```
0  0  0  0  0  0
0  0  0  0  0  1
0  0  0  0  1  0
0  0  0  0  1  1
0  0  0  1  0  0
         ⋮
1  1  1  1  1  1
```

のように0から63までの数の2進数表示が出てきます。この0と1が右からA, I, U, E, O, Kの値（すなわちアヤコさん, イクコさん, ……が海へ行くか行かないか）になっています。

●事情（条件）の表わし方

64通りの組合せのうち, 事情によって成り立たない組合せを取り除いていきます。そうして最後に残ったものが可能性のある組合せということになります。

6人のうち最低3人いないとだめであるということ（条件1）を表わすことを考えます。条件1が成り立たないとき行番号αの行へ行く, という書き方にすると

```
IF A+I+U+E+O+K<3 GOTO α(または, THEN α)
```

のようになります。こうすればA, I, U, E, O, Kのうち少なくとも3つが1ならばαへ行かず下へ抜けることになります。

アヤコさんとカズオ君はけんかをしていて, どちらか一方だけが海へ行く（条件2）ので, AとKが同じになった場合を除けばよく,

IF　A＝K　GOTO　α

となります。(あるいはIF　A＋K＜＞1　GOTO　αでもかまいません)

イクコさんとエイコさんがいっしょ（条件3）というのはこの逆で

IF　I＜＞E　GOTO　α

と表わせます。(あるいはIF　I＋E＝1　GOTO　αでもかまいません)。

オウイチ君はウーちゃんが行くなら行く（条件4）というのは，

IF　U＜0　GOTO　α

となります。

イクコさんが行くときを除いてはカズオ君とウーちゃんはいっしょにはいかない（条件5）ということは，同時にI＝0，K＝U＝1とはならないということ，つまり（I－1)＊K＊U≠0ということで，

IF（I－1)＊K＊U＜＞0　GOTO　α

と表わせます。

アヤコさん，イクコさん，ウーちゃん，エイコさんのうち1人は行かない（条件6）というのは

IF　A＊I＊U＊E＝1　GOTO　α

となります（IF　A＋I＋U＋E＝4　GOTO　αとも書けます)。

次は，アヤコさんかオウイチ君のうち1人は必ず行く（条件7），ウーちゃん，オウイチ君，カズオ君のうち1人は必ず行く（条件8）というのはおのおの，

IF　A＋O＝0　GOTO　α

IF　O＋U＋K＝0　GOTO　α

となります。

最後に，アヤコさんだけが行かないということはない（条件9）は

IF　（1－A)＊I＊U＊E＊O＊K＜＞0　GOTO　α

となります。

このように条件をそのままIF文で書いてやればいいのです。IF文で除かれずに残った組合せだけがPRINT文のところにきて印刷されます。

●論理パズル（海へ行く問題）のプログラム

プログラムは**図1-1**のようになります。これを動作させると**図1-**

```
010 PRINT "ﾛﾝﾘﾊﾟｽﾞﾙ---ｳﾐ ﾍ ｲｸ"
020 DIM I$(2)
030 READ I$(0),I$(1)
040 DATA "ｲｶﾅｲ "," ｲｸ  "
050 FOR A=0 TO 1
060  FOR I=0 TO 1
070   FOR U=0 TO 1
080    FOR E=0 TO 1
090     FOR O=0 TO 1
100      FOR K=0 TO 1
110       IF A+I+U+E+O+K<3 GOTO 220
120       IF A=K GOTO 220
130       IF I<>E GOTO 220
140       IF U<O GOTO 220
150       IF (I-1)*K*U<>0 GOTO 220
160       IF A*I*U*E=1 GOTO 220
170       IF A+O=0 GOTO 220
180       IF U+O+K=0 GOTO 220
190       IF (1-A)*I*U*E*O*K<>0 GOTO 220
200       PRINT "ｱﾔｺ  ｲｸｺ  ｳｳﾁﾔﾝ ｴｲｺ ｵｳｲﾁ ｶｽﾞｵ"
210       PRINT I$(A);I$(I);I$(U);I$(E);I$(O);I$(K)
220       NEXT K
230      NEXT O
240     NEXT E
250    NEXT U
260   NEXT I
270  NEXT A
280 END
```

図1-1 論理パズル（海へ行く問題）のプログラム

```
ﾛﾝﾘﾊﾟｽﾞﾙ---ｳﾐ ﾍ ｲｸ
ｱﾔｺ   ｲｸｺ   ｳｳﾁﾔﾝ ｴｲｺ ｵｳｲﾁ ｶｽﾞｵ
 ｲｸ   ｲｶﾅｲ  ｲｸ   ｲｶﾅｲ  ｲｸ   ｲｶﾅｲ
```

図1-2 論理パズルプログラムの出力結果

2が出てきます。結局「アヤコさんとウーちゃんとオウイチ君が海へ行くことになった」というわけです。

プログラムにむずかしいところはなにもありませんが，「行かない」，「行く」というメッセージを文字列型配列I$に入れているところに注意して下さい。こうすると結果の表示が簡単にきれいにできます。こうしないとIF文やGOTO文が必要になり，面倒になってしまいます。

●字下げ

字下げ（インデンテーション）とは，次のようにFOR～NEXTの中身を右にずらすことです。

```
     ……
FOR  A=…
       ：     } 中身
  NEXT  A
     ……
```

こうするとプログラムが見やすくなります。FOR～NEXTが何重にも重なったとき（入れ子になったとき），その中身が一目でわかります。

字下げはしなくてもプログラムは動きますが，そのプログラムを大事にしたいならば，ぜひ字下げをしてください。そうすればあとでそれを読むとき，とてもやりやすくなります。

NEXTの行を字下げする方法と，しない方法がありますが，この本ではNEXTは字下げすることにしました。

●変数名のつけ方

変数の名前について考えてみましょう。変数名はほかの変数と区別できるものならばなんでもよいわけですが，わかりやすいプログラムにするには，よく考えて名を付けるのがよいのです。図1-1のプログラムではアヤコさん，イクコさん，……が海へ行くかどうかを表わす変数は，その頭文字をとっておのおの，A，I，……としています。このように変数の表わすものと合った名がよいのです。

基本BASICと違い，拡張されたBASICでは2字文以上の長い変数名が許されているものもあります。そのようなBASICが使えるときには，AYAKO，IKUKOなどの長い変数名を使った方がもっとプログラムがわかりやすくなります。

●少し一般的な論理パズル・プログラム

図1-1のプログラムは，1つのパズルが解けるだけで，ほかのパズルが解けるわけではありません。出てくる人も6人に限られています。そこで，もう少し柔軟なプログラムを考えてみましょう。変数A，I，

```
1010 PRINT "ｲﾛｲﾛﾅ ﾛﾝﾘﾊﾟｽﾞﾙ ｵ ﾄｸ"
1020 PRINT "    2000ｷﾞｮｳ ｶﾗ 2999ｷﾞｮｳ ﾆ"
1030 PRINT "    ｼﾞｮｶﾞｲ ｽﾙ ｼﾞｮｳｹﾝ ｵ ｲﾚﾃ"
1040 PRINT "    ｸﾀﾞｻｲ。 ｹｲｼｷ ﾊ,"
1050 PRINT "     IF <ｼﾞｮｳｹﾝ> THEN 3040"
1060 PRINT "    ﾃﾞｽ。"
1070 PRINT "     DELETE 1020-1090 ﾄ ｼﾃ"
1080 PRINT "    ｶﾗ RUN ｼﾃ ｸﾀﾞｻｲ。"
1100 DIM X(20),I$(2)
1110 READ I$(0),I$(1)
1120 DATA  "X",  "O"
1130 PRINT "ﾓﾉ ﾉ ｶｽﾞ";
1140 INPUT N
1150 FOR N1=1 TO N
1160     X(N1)=0
1170     NEXT N1
1180 FOR T=1 TO 2^N
2000 IF X(1)=X(2) THEN 3040
2010 IF X(1)+X(2)+X(3)<2 THEN 3040
3000     PRINT "ﾊﾞﾝｺﾞｳ X ｶ O"
3010     FOR N1=1 TO N
3020         PRINT N1;I$(X(N1))
3030         NEXT N1
3040     FOR N1=1 TO N
3050         IF X(N1)=1 THEN 3080
3060         X(N1)=1
3070         GOTO 3100
3080         X(N1)=0
3090         NEXT N1
3100    NEXT T
3200 END
```

図 1-3 少し一般的な論理パズルプログラム

U, E, O, Kの代わりに,

X(1), X(2), ……, X(N)

という配列の要素を使えば少し一般的に表わせます。図 1-1 のプログラムの何重ものFOR～NEXT入れ子の代わりに, 次のような一重のFOR～NEXTループで表わすことができます。またここではパズルに出てくる人数(N)も自由に選ぶことができます。

```
    ⋮        (配列Xはクリアされているものとする)
INPUT N
FOR T=1 TO 2^N
  ┌──────────────┐
  │ひとつの場合の調査│
  └──────────────┘
```

```
FOR N1=1 TO N
    IF X(N1)=1 THEN
    X(N1) =1
    GOTO
    X(N1)=0
    NEXT N1
NEXT T
    ⋮
```

Xを2進数とみて1をたすことを行なっている

こうして作ったのが**図1-3**のプログラムです。条件を表わすIF文は後から入れるようになっていて，そのようにせよというメッセージが出て，それに従えばよいようになっています。

課　題

別の条件にしてみましょう。今度はどうなるでしょうか。また例の6人に登場してもらいます。

頭で考えて解く方法，プログラムを作って解く方法，また図1-3のプログラムを使って解く方法もあります。答を出してみてください。

1—1　やはり6人が海へ行きますが，今度の事情は次のようなものです。

まず，2人ずつ組になったとき1人あぶれるといけないので，海へ行く人数は偶数でないと困ります。

女の子も男の子も2人はいないと楽しくないのです。

アヤコさんはイクコさんが行くなら行きます。ウーちゃんはエイコさんが行くなら行きます。オウイチ君はカズオ君が行くなら行きます。

イクコさん，エイコさん，カズオ君のうち1人は残らなければなりません。(解答182ページ)

1—2　また6人が海へ行きますが，今度はイカダに乗って川を下って行きます。ところがイカダにはちょうど100kg分しか乗れませ

ん。それより重くするとイカダは沈んでしまうし，軽いと不安定になりひっくり返ります。アヤコさん，イクコさん，ウーちゃん，オウイチ君，カズオ君の体重はそれぞれ，10，20，30，40，50，60kgで，また人以外の物はイカダに乗せないとします。

アヤコさんはイクコさんが行くなら行きます。ウーちゃんはエイコさんが行くなら行きます。オウイチ君はカズオ君が行くなら行きます。

問題は少し違っています。以上の条件で必ず行くことになるのはだれでしょうか，それから必ず残ることになるのはだれでしょうか，というのが問題です。今度は組合せは1通りにはならないのです。

(解答182ページ)

休憩室 1

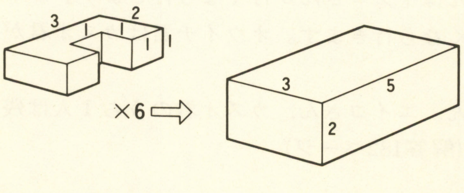
この問題はコの問題

6個のコの字型ペンタキューブ（立方体を5個つなげたたもの）を組み合わせて直方体を作ってください。

ヒント： 全体の体積は6×5＝30ですから，直方体の縦，横，高さは，

5×3×2

になります。これ以外はできません。 (解答208ページ)



2★論理パズル2

本を並べたり お見合いをさせたり

【問　題】

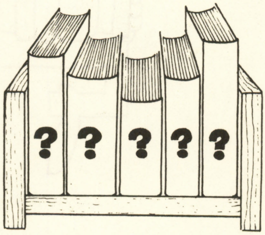
本棚に５冊の本が並んでいます。どんな順番に並んでいるのでしょうか？

「乱調文学大辞典」と「新喜劇思想大系」は隣り合っていません。

この本（あなたが今読んでいる本）と「つつみがみっつ」は隣り合っています。この本は，端にあります。「つつみがみっつ」は，端にありません。

「ナルニア国物語」は中央から見て左にはありません。

「新喜劇思想大系」は中央にありません。



今度の問題も論理パズルですが，前の海へ行く問題とは少し違います。前はある人をとってみれば，「海へ行く」と「海へ行かない」の２通りだったわけですが，この問題では，１つの本がどこにあるかというのは５箇所の可能性があります。さらに２つの本が同じところにくることはないのですから，プログラムは少し面倒になってきます。

●頭と紙で解いてみる

本棚のパズルを解くには，表を作ってみるのがよいでしょう。

表2-1を見てください。おのおのの条件がここに書かれてあります。×印は置けない場所を表わします。この表の中に５個の○を書き，１本の縦の列や横の列に，○が１個しかこないようにすればよいのです。なぜなら１つの場所に２つの本があったり，１つの本が２つの場所に

表2-1　本棚のパズルを解く表

	左端		中央		右端	
乱調文学大辞典						隣り合わない
新喜劇思想大系			×			（隣り合わない）
この本		×	×	×		隣り合う
つつみがみっつ	×				×	（隣り合う）
ナルニア国物語	×	×				

あったりはしないからです。

ここで「乱調文学大辞典」と「新喜劇思想大系」が隣り合わないこと,「この本」と「つつみがみっつ」が隣り合うということも使います。この結果は**表2-2**のようになります。これ以外にはありません。

表2-2　本棚パズルの答

	左				右
乱調文学…			○		
新喜劇…			×		○
この本	○	×	×	×	
つつみが…	×	○			×
ナルニア…	×	×		○	

●すべての可能性を考える

今度は, とり得る全部の場所とはどんなものでしょうか。５個のものを重ならないように５箇所に割り当てるわけです。最初の１個は５箇

所だから５通りあり，次の１個は最初の１個がある場所を除いた残りだから４通りとなり，その次は前の２つのある場所を除くから３通りで……となるので結局

５×４×３×２×１＝５！＝120通り

あることになります。つまり５個から５個取り出す順列ということになります。

この120通りを全部ためすプログラムを書いてみましょう。

●順列をプログラムする

問題のプログラムは，**図2-1**のようになります。このうち順列を作り出している部分は行番号110～220です。

ここで行なうことは，配列要素X(1)，X(2)，……，X(5)に１，２，……，５をすべての組合せで代入することです。R(i)（i＝１，２，……，５）という配列要素は，残っている数字のうちのR(i)＋１番目のものをX(i)に入れることを表わします。例で説明してみましょう。Nが107だとすると，

N＝４×４！＋１×３！＋２×２！＋１×１！

ですから，R(1)＝４，R(2)＝１，R(3)＝２，R(4)＝１となります。まず，１，２，３，４，５の４＋１＝５番目つまり５をX(1)とします，５のところには代りに１を入れておきます。次には残りの２，３，４，１の１＋１＝２番目つまり３をX(2)とします。そのところには２を入れておきます。その次には残った２，４，１の２＋１＝３番目つまり１をX(3)とします。残りは４，２となります。この１＋１＝２番目つまり２がX(4)となります。X(5)は４です。結局，X(1), X(2), ……, X(5)は５，３，１，２，４となります。

Nを０から５！まで変えるとR(1)は０から４，R(2)は０から３，R(3)は０から２というように変わり，これが残った数の何番目を取るかを指定しているわけですから，X(1), ……, X(5)には全部の順列ができあがるというわけです。

FOR～NEXTの入れ子などによる１章と同じ方法を使って組合せが重なるのを除く方法がすぐ考えられますが，あまり薦められません。

```
0010 PRINT "HON NO NARABIKATA"
0020 DIM X(5),R(5),J$(5),D$(5)
0030 FOR M=1 TO 5
0040    READ J$(M),D$(M)
0050    NEXT M
0060 DATA "ﾗﾝﾁｮｳﾌﾞﾝｶﾞｸﾀﾞｲｼﾞﾃﾝ","ｲﾁﾊﾞﾝ ﾋﾀﾞﾘ"
0070 DATA "ｼﾝｷｹﾞｷｼｿｳﾀｲｹｲ      ","ﾋﾀﾞﾘ ｶﾗ 2ﾊﾞﾝﾒ"
0080 DATA "ｺﾉ ﾎﾝ             ","ﾏﾝﾅｶ"
0090 DATA "ﾂﾂﾐ ｶﾞ ﾐｯﾂ        ","ﾐｷﾞ ｶﾗ 2ﾊﾞﾝﾒ"
0100 DATA "ﾅﾙﾆｱｺｸﾓﾉｶﾞﾀﾘ      ","ｲﾁﾊﾞﾝ ﾐｷﾞ"
0110 FOR N=0 TO 119
0120    R1=N
0130    FOR M=1 TO 5
0140       R(6-M)=R1-INT(R1/M)*M
0150       R1=INT(R1/M)
0160       X(M)=M
0170       NEXT M
0180    FOR M=1 TO 5
0190       W=X(M)
0200       X(M)=X(M+R(M))
0210       X(M+R(M))=W
0220       NEXT M
0230    IF ABS(X(1)-X(2))=1 THEN 330
0240    IF ABS(X(3)-X(4))<>1 THEN 330
0250    IF (X(3)-1)*(X(3)-5)<>0 THEN 330
0260    IF (X(4)-1)*(X(4)-5)=0 THEN 330
0270    IF X(5)<3 THEN 330
0280    IF X(2)=3 THEN 330
0290    PRINT "      HON                 BASHO"
0300    FOR M=1 TO 5
0310       PRINT "     (";J$(M);",";D$(X(M));")"
0320       NEXT M
0330    NEXT N
0340 END
```

図2-1　本棚のパズルのプログラム

繰返しの数は5^5＝3125通りも調べなければなりません。120に比べ多すぎます。本の数が多くなればほとんど不可能になるでしょう。

ここで述べた順列の作り方のほかにもいろいろな方法があり，出てくる組合せの順序がおのおの違っています。みなさんも順列の作り方を考えてみてください。

●プログラムの説明

X(M)という配列要素はMという本のある場所を表わしています。「本」というものと「場所」というものを逆にしても表わせますが，こ

の場合はこのようにしました。

またJ＄(M)はMという本の名前，D＄(X)は場所Xを表わす言葉です。1章でもいいましたが，こういう配列を使うとわかりやすくなります。

IF文のところでは1章と同じく除外する条件を試しています。

ABS(X(1)−X(2))＝1

ということは，X(1)（つまり本1の位置）とX(2)（つまり本2の位置）の差が1ということですから，本1（乱調文学大辞典）と本2（新喜劇思想大系）が隣り合っているということを表わすことになります。また，

(X(3)−1)＊(X(3)−5)<>0

の意味は次のようになります。1や5というのは端の場所です。X(3)は1にも5にもならないということですから，「本3（この本）は端にない」ということになります。

以上で問題の内容がうまくIF文の中身に訳されているのがわかると思います。

プログラムの結果を見てみましょう。**図2-2**のようになります。

```
HON NO NARABIKATA
    HON                        BASHO
  (ﾗﾝﾁｮｳﾌﾞﾝｶﾞｸﾀﾞｲｼﾞﾃﾝ,ﾏﾝﾅｶ)
  (ｼﾝｷｹﾞｷｼｿｳﾀｲｹｲ       ,ｲﾁﾊﾞﾝ ﾐｷﾞ)
  (ｺﾉ ﾎﾝ               ,ｲﾁﾊﾞﾝ ﾋﾀﾞﾘ)
  (ﾂﾂﾐ ｶﾞ ﾐｯﾂ          ,ﾋﾀﾞﾘ ｶﾗ 2ﾊﾞﾝﾒ)
  (ﾅﾙﾆｱｺｸﾓﾉｶﾞﾀﾘ        ,ﾐｷﾞ ｶﾗ 2ﾊﾞﾝﾒ)
```

図2-2　本棚のパズルのプログラムの出力結果

●男女の組を作る

本棚のパズルの本質は「n個のものをn個の場所に割り当てる」ということです。これと同じになる問題はいっぱいあります。男女を組み合わせる問題というのがありますが，これもそうです。解いてみましょう。

【問　題】

イチカワヒトミ，ニイヤマフミ，ミタミミ，ヨシダヨシミ，ゴウイツミの5人の女性と，イチカワカズオ，ニイヤマジロウ，ミタサブロウ，ヨシダヨシオ，ゴウゴロウの5人の男性がいて集団見合いをすることになりました。ただし，

- 同じ姓の男女はきょうだいである。
- サブロウはヒトミとフミを気に入っていて，どちらかと付き合いたいと思っている。
- ヒトミはフミにライバル意識を持っていて，フミがサブロウと付き合うなら，自分はサブロウの友人で風采のあがらないゴロウとは付き合いたくないと思っている。
- ゴロウとミミは別れたばかりである。
- ヒトミはヨシオが嫌いである。
- イチカワ家とゴウ家は親戚になりたいと考えている。
- ニイヤマ家とヨシダ家はけんかしている。

どのような組合せで見合いをすれば無難でしょうか？

プログラムの形式は本棚のパズルと同じなので，条件判定のところだけ説明します。

イチカワ，ニイヤマ，……の順で1，2，……，5と番号付けをすることにし，X(M)はM番目の男性が見合いする女性の番号ということにします。すると同じ姓の男女が見合いする場合を除くのは，次のようなFOR～NEXTで表わせます。

```
FOR  M=1  TO  5
     IF  X(M)=M  THEN  α
     NEXT  M
```

さらに後の部分は特別なことは必要なく，

```
IF  (X(3)-1)*(X(3)-2)<>0  THEN  α
IF  ABS(X(3)-2)+ABS(X(5)-1)=0  THEN  α
IF  X(5)=3  THEN  α
```

```
IF  X(4)＝1  THEN  α
IF  (X(1)－5)＊(X(5)－1)<>0  THEN  α
IF  (X(2)－4)＊(X(4)－2)＝0  THEN  α
```

と書けばよいのです。αは次の組合せに進むためのNEXT文の文番号です。

課　題

いろいろな問題を出してみます。プログラムを作ってみてください。

2－1　登場人物は先ほどの10人です。5人の女性の年令は高い方からヒトミ，フミ，ミミ，ヨシミ，イツミの順，男の方もカズオ，ジロウ，サブロウ，ヨシオ，ゴロウの順です。

今度はヨシオとゴロウが双子，ヨシミとイツミが双子で，ヨシオとゴロウは見合いの相手は双子同士か2人とも双子でないかがよいと思っています（つまり双子の一方が双子の一方だけと見合いはしたくない）。

ヨシミの相手はイツミの相手より年をとっていないと困ります。

ヒトミ，フミ，ミミ，ジロウの4人はきょうだいです。

ヒトミ，フミ，ミミは相手の年令が自分たちの年令と同じ順になっていてほしいと考えています。

ヨシミはカズオが嫌いです。

以上の条件ではどのように組み合わせたらよいでしょうか。（解答183ページ）

2－2　一郎，二郎，三郎，四郎，五郎，六郎の兄弟がこの順で並んで座っていましたが，みんなは隣りに座っている兄や弟とけんかを始めました。

もうけんかをしないように，けんかをしていた者同士は，隣り同士にも1人隔てて隣りにもならないように，座り直させたいと思います。どうすればよいでしょうか。(解答184ページ)

２－３　１から６までの６個の数字を１cmの間をおいて一直線に並べます。ただしどの２つの数字をとっても，その数の差と，その間の距離が同じにならないようにしてください。

たとえば，

１　６　２　３　５　４

ならば，１と５の間の距離が４（＝５－１）となってだめです。どうすればよいでしょうか。(解答184ページ)

２－４　課題２－２や２－３で人の数や数字の数が多くなったらどうなるでしょうか。(解答186ページ)

２－５　課題２－２や２－３で直線に並べるのではなく，両端をつないで環状に並べたらどうなるでしょうか。(解答187ページ)

２－６　アメリカ，イギリス，ウガンダ，エチオピア，オーストラリアの代表がこの順で並んで討議していましたが，席を取り替えることになりました。ただし元いた席には座らないことにします。ウガンダは中央より左に座ります。ウガンダが左から２番目に座ったときには，オーストラリアは孤立するといやなので左端には座りません。オーストラリアは中央に座るほどの大国ではないといっています。エチオピアは左端が嫌いです。アメリカとオーストラリアはどちらかが元いた座に他方を座らせるでしょう。イギリスとエチオピアは，今度は中央か端がいいといっています。(解答188ページ)

２－７　課題２－２，２－３，２－４では，左右が裏返しであることを除けば同じである解がでてきてしまいました。これを除くにはどうすればよいでしょうか。(解答188ページ)

２－８　課題２－５では，並びが環状になっているので，１つの解をずらしてもまた解になってしまいます。これを除くには，どうすればよいでしょうか。(解答188ページ)

休憩室 2

変数名はよく考えて付けよう

変数名は，あとで読んでもよくわかるようなものにしましょう。そのためには

(1) 変数名をその意味に合ったものにする

(2) 規則的に名前を付ける

ということが大事です。基本BASICでは，変数名は英字1字または英字の次に数字を並べたものということになっています。ですから長い名前を付けることはできません。しかし，座標ならXとY，桁を表わすならKというように，なるべく想像のつく名前にしましょう。強力なBASICではたいてい長い名前が付けられます。その場合には桁ならばKETA，スタックならSTACK，盤の配列ならBANというように付けるとよいでしょう。

規則的に名前を付けるということも大事です。たとえば，1からDまでFORループを回すなら

```
FOR  D1＝1  TO  D
```

1からXなら

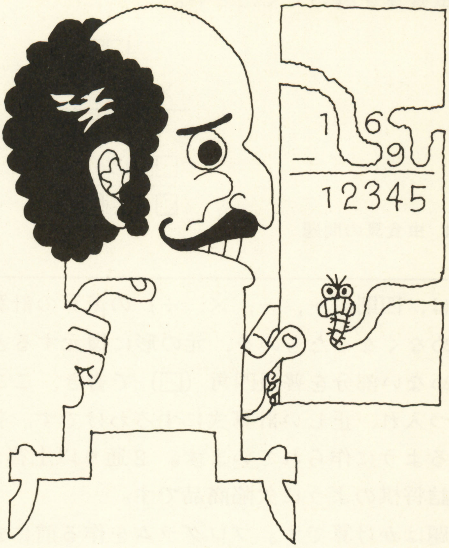
```
FOR  X1＝1  TO  X
```

というように，制御変数とFORの終値を対応付けます。

また，スタックS1のスタックポインタはP1，スタックS2のものはP2，というように付けます。

このようなやり方は，人によって違っていてもかまいませんが，いつも同じようにするのがよいのです。

3★虫食算を解く



【問　題】

次の虫食算を解いてください。

```
    □□
×   □□
──────
    □1
  □□2
──────
  □4□□
```

図3-1　虫食算の問題

虫食算は，四則（＋，－，×，÷）の筆算の計算式の一部が虫に食われて読めなくなったものを，元の形に復元するというのがはじまりです。読めない部分を普通四角（□）で書き，ここに０から９までの数字を１つ入れ，正しい計算式にするわけです。普通虫食算の答は１通りになるように作られています。２通り以上答がある虫食算は，余詰のある詰将棋のように欠陥商品です。

この問題はかけ算です。プログラムを作る前にまず人間がどう虫食算を解くかを，この問題を例に取って述べてみます。

●人間はどう虫食算を解くか

始める前に，どの四角に数を入れるかということを述べやすくするために，**図3-2**のように四角に名前を付けておきます。これはプログラムを作るときにもそのまま使うので，よく覚えておいてください。問題は$x \times y = u$の形で，中間結果はz_1，z_2で表わします。またxの一の位，十の位をそれぞれX(1)，X(2)のように表わします。

ではこの問題を解いてみましょう。解き方はいろいろあり，人によって考える筋道は違いますが，ここではz_1の一の位Ｚ（１，１）が１であることに着目してみましょう。このことはxの一の位X(1)とyの一の位Y(1)の積，

X(1)×Y(1)

の一の位が１であるということです。九九を思い出せばわかるように，

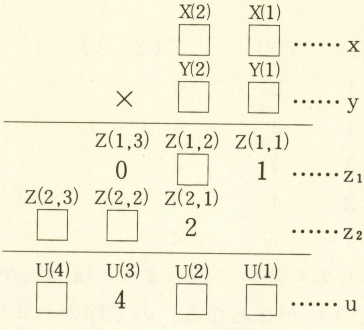


図 3 - 2　数字の位置に名前を付ける

この組合せは,

1 × 1

3 × 7

7 × 3

9 × 9

の４通りがあることがわかります。ここで,

$x \times Y(1) = z_1$

$x \times Y(2) = z_2$

で，z_1 は２桁，z_2は３桁ですから，Y(2)はY(1)より大きいということがわかります。したがって，９より大きい数字はなく，上の９×９の場合は除かれます。

次にz_2の一の位Z（２，１）が２であることに着目してみます。このことはX(1)×Y(2)の一の位が２ということです。X(1)には１，３，７の可能性がありますから，この組合せは,

1 × 2

3 × 4

7 × 6

の３通りが考えられます。

U(3)が４ですから，繰り上がりを考えてもZ（２，２）は３か４です。このことからX(2)を調べてみます。x×Y(2)の組合せは次のようなものが出てきます。

	x			
	X(2)	X(1)	Y(2)	Z(2，2)
(a)	2	1	2	4
(b)	7	1	2	4
(c)	3	3	4	3
(d)	8	3	4	3
(e)	5	7	6	4

このうちだめなものを除いていきます。(a)はz_2が２桁になってしまう（つまりZ（２，３）が０となる）ので除かれます。

(c), (d), (e)はz_1が３桁になってしまうので除かれます。

結局(b)だけが残り，答は**図３-３**のようになります。

```
      7 1
  ×   2 1
  -------
      7 1
  1 4 2
  -------
  1 4 9 1
```

図３-３　解　答

このように人が解く場合は，いろいろなところの手がかりを利用して効率よく分析していきます。たくさんの場合の数を調べるということはしません，手がかりを使って場合の数を数通りにおさえます。むずかしい場合でも数十通りの場合の数をみればよいようです。場合の数が数百通りになるならば，どこかで方針を間違えたか，手がかりを見落としているのです。数百通りになってしまえば，到底人手で処理しきれるものではありません。

●虫食算のプログラムの方針

人間がどう虫食算を解くかということを分析すると，虫食算プログラムを作るための参考になります。しかし人間が解く場合と，プログラムによって解く場合とでは大きな違いがあります。プログラムを用いる場合の長所としては，人間が行なうのに比べてずっと数多くの場

合をチェックできる，ということがあります。人間がせいぜい100通りまでなのに対して，プログラムでは，問題にもよりますが，数十万通りを調べることができます。プログラムを用いる場合の欠点は，人間が解くときのような柔軟な対処をすることがとてもむずかしいということです。人間のように柔軟な対処をさせるためにはプログラムの中に，

① 人間の行なう非常に多くの対処法を書いておく

または，

② 人工知能の高度な手法を取り入れる

ことが必要です。

100行程度のプログラムでは，このことはほとんど不可能なので，全部の場合を調べる，というのを基本として，わかりやすいものを作り，ある程度の工夫をするということに止めたいと思います。

●虫食算プログラムの説明

プログラムを**図3－4**に示します。これを眺めてみましょう。プログラムは次の4つの部分からできています。

① 数値の下からK桁目の値を与える関数

② 問題の入力部

③ 問題

④ 組合せを作り出し，正しい答かどうか調べる部分

この中では④が中心的な処理を行ないます。プログラムは①～④の

```
0010 PRINT "MUSHIKUIZAN"
0020 DIM X(10),Y(10),Z(10,11),U(20)
0030 REM 10^(K-1) NO KETA
0040 DEF FNA(W1)=W1-INT(W1/10)*10
0050 DEF FNK(W,K)=FNA(INT(W/10^(K-1)))
0060 READ M,N
0070 FOR M1=1 TO M
0080   READ X(M+1-M1)
0090   NEXT M1
0100 FOR N1=1 TO N
0110   READ Y(N+1-N1)
0120   NEXT N1
0130 FOR N1=1 TO N
0140   FOR M1=1 TO M+1
```

```
0150      READ Z(N1,M+2-M1)
0160      NEXT M1
0170    NEXT N1
0180 FOR L=1 TO M+N
0190   READ U(M+N+1-L)
0200   NEXT L
0210 DATA 2,2
0220 DATA            -1,-1
0230 DATA            -1,-1
0240 REM -----X--------------
0250 DATA          0,-1, 1
0260 DATA       -1,-1, 2
0270 REM --------------------
0280 DATA       -1, 4,-1,-1
0290 REM
0300 REM KEISAN
0310 FOR X9=10^(M-1) TO 10^M-1
0320   FOR M1=1 TO M
0330     IF X(M1)=-1 GOTO 350
0340     IF FNK(X9,M1)<>X(M1) GOTO 580
0350     NEXT M1
0360   FOR Y9=10^(N-1) TO 10^N-1
0370     FOR N1=1 TO N
0380       IF Y(N1)=-1 GOTO 400
0390       IF FNK(Y9,N1)<>Y(N1) GOTO 570
0400       Z9=FNK(Y9,N1)*X9
0410       FOR M1=1 TO M+1
0420         IF Z(N1,M1)=-1 GOTO 440
0430         IF FNK(Z9,M1)<>Z(N1,M1) GOTO 570
0440         NEXT M1
0450       IF Z(N1,M+1)<>-1 THEN 470
0460       IF FNK(Z9,M+1)=0 THEN 570
0470       NEXT N1
0480     U9=X9*Y9
0490     FOR L=1 TO M+N
0500       IF U(L)=-1 GOTO 520
0510       IF FNK(U9,L)<>U(L) GOTO 570
0520       NEXT L
0530     IF U(M+N)<>-1 THEN 550
0540     IF FNK(U9,M+N)=0 THEN 570
0550     REM DEKIAGARI
0560     PRINT "KOTAE   ";X9;" * ";Y9;" = ";U9
0570     NEXT Y9
0580   NEXT X9
0590 GOTO 60
0600 END

MUSHIKUIZAN
KOTAE      71  *    21  =  1491
```

図 3-4　虫食算プログラムと結果

ようにわかりやすく分けることが大事です。異なる機能ごとに分けるというのがひとつの基本です。

●問題の入力部と問題の与え方

プログラムのREAD　M，NのところからNEXT　Lのところまでの十数行で問題を入力します。問題はその後にDATA文で与えます。この問題入力部で図3-2のような配列X，Y，Z，Uに値が格納されます。また，そのまえに，Xの桁数MとYの桁数Nが入力されています。このようにXとYはどんな数でもとれるようにしてあり，理論的にはどんなかけ算の虫食算でも解けることになります（ただし，あとで述べるように，桁数が多くなってくると，計算時間の問題が出てきます)。

さて，このDATA文を見てください。一目で問題がわかると思います。このように見やすく書くということは，ひとつの大事な**精神**なのです。1行にベタッと書いても動くかもしれませんが，そういうものは見にくいのです。キーインのミスを誘発したり，またミスが発見しにくくなったりし，結局ソフトウエアの品質が下がったり，完成が遅れたりします。見にくくすることによって計算速度が速くなるという人がいますが，これはほとんど問題になりません。

わかりやすいプログラムを書こう

次に虫食算の問題データの与え方について考えてみましょう。一般にデータの与え方には，次の4通りがありえます。

① INPUT文を使ってキーボードから入力する

② プログラムの中で代入文によって与える

③ プログラムの中でREAD文とDATA文によって与える

④ INPUT＃文などを使ってカセットやフロッピーディスクの中のファイルからデータを取り出して使う

このうち①は一度入力したデータが消えてしまうのでよくありません。②はきたないプログラムになってしまいます（一般に変数に代入させる値を，プログラムの中ではっきりさせたいときに②を使うこと

もありますが)。④は全体が２つ以上のファイルに分かれてしまいます。ファイルはなるべく増えないようにすべきです。年月がたつとファイルの名前や中身が，なんだったかというのはどんどん忘れていきます。特に中年以降の人はそうです。結局③の方法がこの場合はいいでしょう。ただし，もっと複雑な処理を行なう場合は④のように入力ファイルを分けた方が柔軟性からいってよい，ということはあります。

●数値の下からK桁目を与える関数

「数値の下からK桁目を与える」というような機能を処理の中心の部分から取り出すことは大切です。このことによって処理の中心の部分（組合せを作り出し，正しい答かどうか調べる部分）をすっきりと書くことができます。

数値の下からK桁目を与える，という機能は関数にして取り出しました。ほかの場合には機能をサブルーチンにして取り出すこともあります。後の章ではサブルーチンにして取り出すことが多くでてきます。関数名はKETAのKをとってFNKとしました。1949311の下から５桁目は４ですから，

FNK（1949311，５）

の値は４になります。

数Wの下からK桁目の数FNK（W，K）の計算の方法は，WのK－１桁以下を除いたもの，

INT（W/10^（K－１））

を10で割って，余りを取ることによって行ないます。FNAは10で割った余りを出す関数です。

●組合せを作り出し正しい答かどうか調べる部分

問題はx×y＝uの形をしています。xはM桁ですから，xの候補X９を10^{M-1}から10^{M}－１まで（Mが２なら10から99まで）変えて，すべての場合を調べます。また，そのおのおのの場合について，同じようにしてyの候補Y９を10^{N-1}から10^{N}－１まで変えます。

xの候補X９の下からM１桁目はFNK（X９，M１）でえられます。

これと問題に書かれてあるｘの下からＭ１桁目の数値Ｘ(Ｍ１）が同じならばよいのです。同じでなければ，その組合せは失敗ですからNEXT　Ｘ９のところへとんで，次の組合せを調べ始めます。Ｘ(Ｍ１）が－１のときは，そこは虫が食っているのですから，そこには何がはいってもいいわけで，その時点ではパスし，次のマスのチェックに進みます。このような処理を，

x, y　かける数
z　途中の値
u　答

の全部について調べ，おのおののマスに問題の方に－１がはいっているか，候補と問題の値が一致するかすれば，その組合せは問題を満足する解ということになります。

プログラムはFOR～NEXTのループに対して字下げをしてあります。これでループの入れ子がどうなっているかが，わかりやすくなっていることに注意してください。IF文が２個並んだところはいくつかあり，どれも同じことをしています。ここをサブルーチンにまとめられればさらにプログラムがすっきりします。しかし，BASICのサブルーチンには引数がなく，これはうまくいきません（関数は機能が低くて使えません)。この程度になるのは仕方がないでしょう。

Ｘ９のチェックだけがＹ９のFORループの外で行なっています。これで少し計算が速くなります。このほかのチェックはＹ９のFORループの中で行なっています。

この問題で使っている方法は，基本的部分だけを取り出してあり，桁が多くなると計算スピードが遅くなってきます。このことを考えてみましょう。

●一生かかっても答が出ない話

このような虫食算のプログラムの計算スピードは，問題の桁数によります。計算時間は調べる組合せの数に比例すると考えてよいでしょう。

たとえば，3桁×2桁の虫食算ならば，3桁の数は100から999ですから約1000通り，2桁の数は10から99ですから約100通りで，全体で，1000×100＝10万通りとなります(細かい値は考えません)。M桁×N桁の虫食算ならば調べる組合せの数は，

$10^M \times 10^N = 10^{M+N}$通り

になります。

さて，1つの組合せを計算する時間を考えてみましょう。次のようにしてループをまわしてみます。

```
10  INPUT  T
20  FOR  S＝1   TO  T
30        NEXT  S
40  END
```

普通のパソコンでこれを走らせてみます。Tに10000を与えると，プログラムが終了するのに20秒前後かかるようです。つまり，このことは中身のないFORループを1回まわるのに，

$$\frac{20}{10000}=0.002(\text{秒})=2\ \underset{\text{ミリ秒}}{(\text{ms})}$$

かかることになります。中身があるFORループについて考えてみると，その中身にはもう繰返しはありませんから，中身なしに比べて数倍から数十倍の時間で1回計算できるであろうと思われます。

1つの組合せを調べる時間をどんな場合でも50ms（＝1/20秒）としてみましょう。実際の時間はこの倍か半分になるかもしれませんが，これ位の違いはここでは大事ではありません。

そうすると，2桁×2桁の虫食算では，100×100×1/20＝500（秒）＝8（分）かかり，3桁×3桁では1000×1000×1/20＝50000（秒）＝約15時間となります。さらにこれを続けると，

4桁×4桁　約50日
5桁×5桁　約15年
6桁×6桁　約1500年
⋮

となります。これを見ると3桁×3桁の虫食算あたりが限界で，これから先は計算時間からいってまったく不可能です。4桁×4桁というのは組合せが1億通りということです。この限界はパソコンだけのものではありません。どんなに高速の計算機でも6桁×6桁（1兆通り）を調べることは不可能です。

ほかのパズルはどうでしょうか。6人組が海へ行くパズルは$2^6=64$通りでした。このようなパズルでn人組の問題ならば，

$2^n \fallingdotseq 10^{0.3n}$（通り）

の組合せがあることになります。27人組になると1億通りを超えます。こうなると全然計算は不可能になります。

本棚に本を並べる問題では，5冊で5！＝120通りでした。n冊ではn！通りとなります。この場合では10冊の問題で組合せの数はほぼ限界になります。

以上をまとめると**図3-5**のようになります。このように組合せを調べる問題には，計算時間の限界が必ずあるのです。それは100万から1000万通りあたりにあります。虫食算（かけ算）の2つの数の桁の

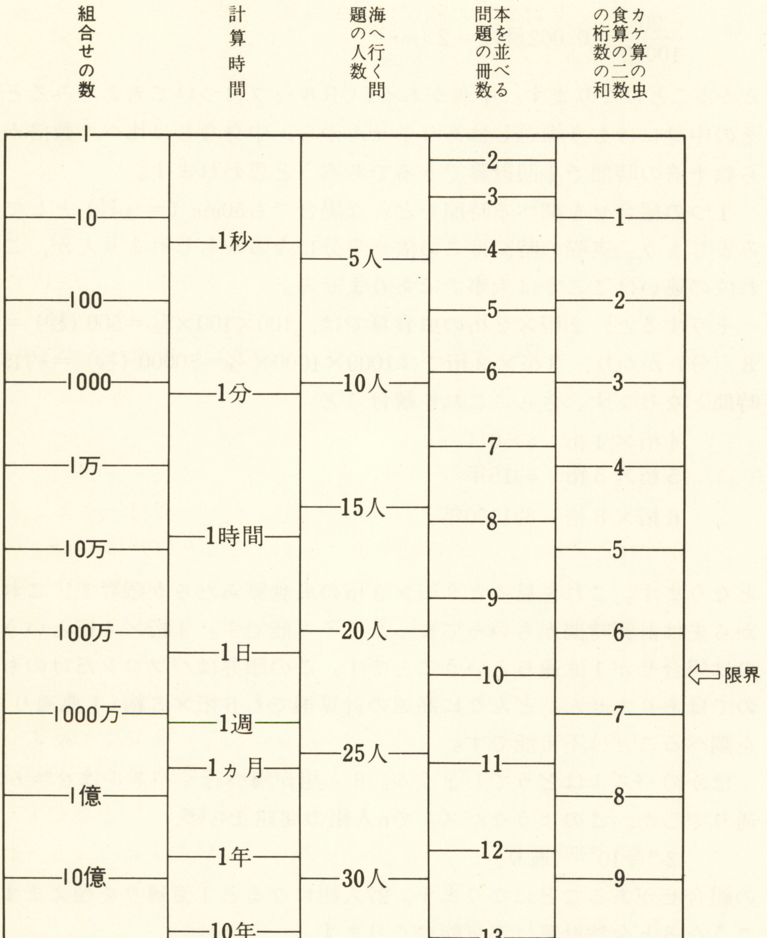


図 3－5　パズルの組合せの数と計算時間

和でいうと７桁あたりということになります。

計算機が100倍速くなったとしてみましょう。これは100万通りが１億通りになるということです。虫食算の限界でいうと７桁が９桁にな

るだけなのです。2数の桁の和が15の虫食算,

```
   ○○○○○○○○○○○
 ×         ○○○○○
──────────────────
          ⋮
```

はまったく不可能なのです。

このように計算が不可能になるのは，桁数などの増加に従って，組合せの場合の数が急に大きくなるからなのです。このことを，組合せ論的爆発（combinatorial explosion）といいます。理論的には答が出てくることになっていても，ディスプレーの前に**一生坐っていてもなにも出てこない**ということもあるわけです。

●虫食算のプログラムと結果

でき上がったプログラムと結果（図3-5）をもう一度見てみましょう。このプログラムは，xの各桁をチェックすること以外は，xとyを変えてできる全部の組合せを作っています。したがって，やはり桁数が増すと計算のスピードは非常に遅くなってしまいます。

●計算スピードを速くする

ここで2桁×2桁の場合を考えてみましょう。これを書くと**図3-6**のようになります。この場合，どのように場合が決められるかというと，**図3-7**のようにx_2，x_1，y_2，y_1の部分が決まって行きます（このようなものを探索木といいます)。この4個の数が決まったら，全

$$
\begin{array}{rrrr}
 & & x_2 & x_1 \\
\times & & y_2 & y_1 \\
\hline
 & z_{13} & z_{12} & z_{11} \\
z_{23} & z_{22} & z_{21} & \\
\hline
u_4 & u_3 & u_2 & u_1
\end{array}
$$

図3-6　2桁×2桁の掛算

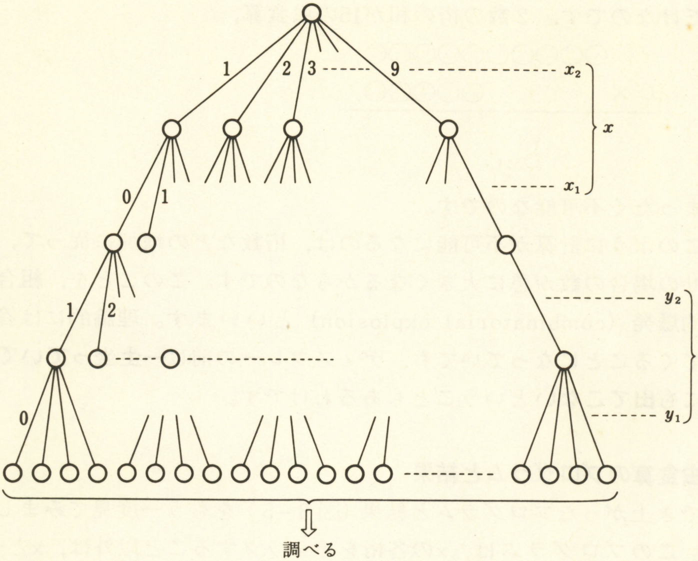


図 3 - 7　探索木

体が正しいかを見ているわけです。

しかし，x_2，x_1，y_2が決まったときに，もうz_{23}，z_{22}，z_{21}のところはチェックできるのです（もちろんy_2も）。

また，決める順序をx_1，y_1，x_2，y_2のようにしたとすると，x_1とy_1を決めたときにもうz_{11}をチェックすることができます。

このようにすると図３－７のような探索木のある枝から先は調べなくてもよいことになります。約100×100＝１万通り調べるところが，xやyを途中まで決めた段階でかなりの数の駄目な場合を除けるので，極端に調べる場合の数が減るはずです。

●途中でチェックするということ

本質は**図３－８**で表わされています。つまり途中でチェックすれば全体の調べる量が減るわけです。ペントミノのような複雑なパズルでは，途中で調べることを自動的に行なうようにプログラムしなければほとんど解けません。

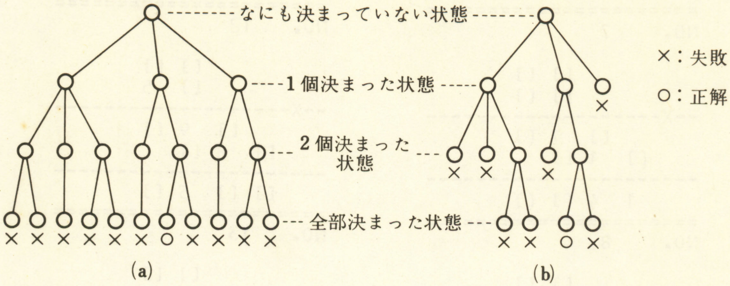


図 3-8　全部の組合せを作って調べる場合(a)と途中でチェックする場合(b)

課　題

3－1　次の虫食算を解いてください。(解答189ページ)

```
========================
NO.    1

            [] []
            [] []
--X---------------------
         [] [] []
      [] []  7
------------------------
      []  8 []  1
========================
NO.    2

             2 []
             4 []
--X---------------------
         []  0 []
      [] [] []
------------------------
      [] [] []  3
========================
NO.    3

            [] []
            []  2
--X---------------------
         [] [] []
      []  4 []
------------------------
      [] []  0  4
```

```
========================
NO.    4

             9 []
            [] []
--X---------------------
         []  7  9
      [] [] []
------------------------
      [] []  0 []
========================
NO.    5

             8 []
            [] []
--X---------------------
         []  7  6
      [] [] []
------------------------
          3 [] [] []
========================
NO.    6

            [] []
            [] []
--X---------------------
         [] [] []
      []  6 []
------------------------
      []  0  2  2
```

```
======================
NO.   7

          [] []
          [] []
--X-------------------
       []  7 []
    []  1 []
----------------------
     1  4 [] []
======================
NO.   8

          [] []
          [] []
--X-------------------
       []  5 []
    []  6 []
----------------------
    [] [] []  9
======================
NO.   9

          []  3
          []  7
--X-------------------
       [] [] []
    [] []  8
----------------------
    [] []  1 []
======================
NO.   10

           5 []
          []  9
--X-------------------
       [] []  6
    []  0 []
----------------------
    [] [] [] []
======================
NO.   11

          []  7
          [] []
--X-------------------
       [] [] []
    []  9  9
----------------------
    [] []  8 []
```

```
======================
NO.   12

          [] []
          []  5
--X-------------------
       []  9 []
    []  8 []
----------------------
    [] []  5 []
======================
NO.   13

          [] []
          [] []
--X-------------------
       []  9  9
    []  8 []
----------------------
    [] [] [] []
======================
NO.   14

          [] []
          []  4
--X-------------------
       [] [] []
    [] [] []
----------------------
     2  5  2 []
======================
NO.   15

          [] []
          []  2
--X-------------------
       []  8 []
    []  2 []
----------------------
    [] []  8 []
======================
NO.   16

          [] []
          []  5
--X-------------------
       []  2 []
    []  7  2
----------------------
    []  1 [] []
```

```
=====================
NO.  17

          [] []
          []  8
--X------------------
       [] [] []
    []  4 []
---------------------
    []  0  1 []
=====================
NO.  18

          [] []
          [] []
--X------------------
       []  2 []
    []  1 []
---------------------
    [] [] []  1
=====================
NO.  19

          [] []
          []  3
--X------------------
       []  7 []
    []  5 []
---------------------
     6 [] [] []
=====================
NO.  20

          [] []
          []  6
--X------------------
       [] [] []
    [] [] []
---------------------
     9  2 [] []
=====================
NO.  21

          [] []
          [] []
--X------------------
       [] [] []
    []  5  5
---------------------
    []  3 []  5
=====================
NO.  22

           9 []
          []  5
--X------------------
       [] [] []
    [] []  0
---------------------
     7 [] [] []
=====================
NO.  23

          [] []
           4 []
--X------------------
       [] []  3
    [] []  6
---------------------
    [] []  0 []
=====================
NO.  24

           6  2
          [] []
--X------------------
       []  5 []
    []  7 []
---------------------
    [] [] [] []
```

3－2　虫食算を作るプログラムを作ってください。方法としては，式の中の場所をでたらめに選びだし，その中に数字を残し，それを解いてみる，というのがやさしいでしょう。(解答189ページ)

3－3　海へ行く論理パズルの計算スピードを速めることを考えてください。たとえば，イクコさんとエイコさんがいっしょに行くというのは，オウイチ君やカズオ君がどうあろうと決まったことです。(解答194ページ)

休憩室 3

0は1を生み，1は2を生み，2は3を生み，……

次のプログラムを動かすとどうなるでしょう。

```
10  A=0
20  A=A+1
30  PRINT A;
40  GOTO 20
50  END
```

これは止まらないですね。ところがこれはどうでしょう。

```
10  A=0
20  A=A+1
30  PRINT A;
40  GOSUB 20
50  END
```

動作をしばらく見ていてください。これは90ページで説明するスタック領域と関係があります。そのようになる理由も考えてみてください。

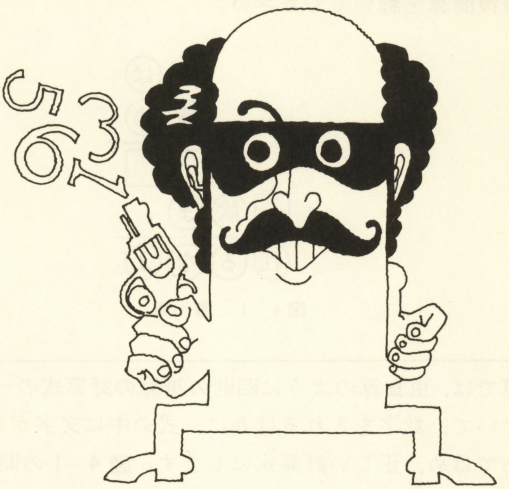
また，行番号40のところを

```
40  RUN
```

とすると，どうなるでしょうか？

(解答208ページ)

4★覆面算を解く



【問　題】

次の覆面(ふくめん)算を解いてください。

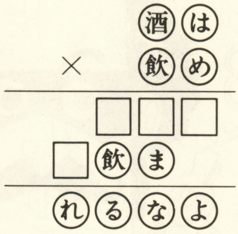


図4－1　覆面算

覆面算では，虫食算のように四則の筆算の計算式の一部が四角（□）になっていて，数字を入れるほかに，式の中に文字があって，それに数字をあてはめ，正しい計算式にします。**図4－1**の問題では，「(酒)，(は)，(飲)，…」が文字で，これに数字をあてはめるわけです。虫食算のときと同じように，答が1通りしかないのが正しい覆面算です。

この問題はかけ算です。覆面算には，虫食算と違って足し算や引き算にもおもしろいものがいろいろあります。

●覆面算の規則

虫食算のときには，四角（□）の中には0から9までのどの数字も入れられました(もちろん1番上の桁に0を入れてはいけませんが)。虫食算の文字にあてはめる数字も0から9までのどれでもいいのですが，次のような制限があります。つまり，1つの式の中で，

① 同じ文字には同じ数字をあてはめる

② 違う文字には違う数字をあてはめる

ということが必要です。

図4－1の問題では，(酒)のところにはいる数字と，(は)にはいる数字とは違わなければなりません。それから(飲)という文字は，2箇所に出てきますが，両方に同じ数字がはいるのです。

これ以外は虫食算と同じです。□にはなにを入れてもいいのです。もちろん1番上の桁の㊇，㊇，㊇には0ははいりません（形からいって㋱にも0ははいりません）。

覆面算のプログラムは，このような規則をうまく表わすことがポイントになります。あとの部分は虫食算のプログラムと同じと考えていいでしょう。

●覆面算プログラムの方針

調べる方法は虫食算と同じように，全部の数の組合せを作り出すようにし，順番に正しい答かどうかチェックしていくことにします。

覆面算プログラムでは，ある数字が1つの場所にあてはめられるかチェックする部分が重要となります。この部分は虫食算の場合に比べてかなり複雑になります。この部分以外は虫食算で行なったのと同じようにできるでしょう。では，このあてはめのチェックを分析してみましょう。

覆面算の1つの場所（位置）にくるものには，

①　覆面の記号（漢字，仮名，アルファベットなど）

②　虫食の枠（□）

③　すでにわかっている数字

の3種が許されるものとします。

この3種に対する処理を考えればいいのです。今，ひとつの組合せをチェックしているものとします。そしてその場所にあてはめようとしている数字をnとします。

その場所が②の場合は，nをそのままそこに入れればいいのですから，いつもあてはめは成功ということになります。1番上の桁に0を入れてしまうという失敗は，後で別に調べます。

その場所が③のとき，その場所にある数字とnが同じなら成功，違っていれば失敗となります。このあたりまでは，虫食算のときと同じです。

その場所が①のとき，すなわち覆面記号のときはいろいろな可能性があります。まず，

（1－1） その覆面記号にすでにほかの場所で数が割り当てられている場合

（1－2） その覆面記号にはまだ数が割り当てられていない場合

の２種類に分けられます。

（1－1）では，その割り当てられている数とnが同じならあてはめは成功となるし，違えば失敗になります。

（1－2）では，数字nがほかの覆面記号に割り当てられていないかどうかで，さらに２種に分かれます。すでに割り当てられていれば，違う記号に同じ数字を割り当てられないので，ここではあてはめは失敗になります。数字nがまだほかで割り当てられていなければ，ここでそこの覆面記号にnを割り当て，結果としては成功ということになります。

以上をまとめると，**図4－2**のようになります。

●覆面算プログラム

まず問題のデータの与え方を考えます。覆面の記号，虫食の枠，出ている数字を区別する必要があります。ここでは**表4－1**のように表わす約束をします。**表4－2**のように文字と数値を対応させると，問題は**図4－3**のようになります。これをデータ文で与えます。覆面の記号は互いに違う数字を表わすので，覆面の記号の数は10個以下です。したがって，－1から－10まで十分表わせるわけです。

覆面算の場合，数字をある場所にあてはめられるかどうかをチェックするというのが，複雑になるといいました。チェックするために２つの配例ＦとＬを用意します。

配列要素Ｆ（α）にはα番目の覆面記号（DATA 文のところでは$-\alpha$で表わされています）に，数がすでに割り当てられたかを示す情報を入れておきます。Ｆ(α) が－１ならばまだなにも割り当てられていません。０以上なら割り当てられています。

配列要素Ｌ(k）には，数字kがどれかの覆面記号に割り当てられているかを示す情報を入れておきます。Ｌ(k) が－１ならばまだなにも割り当てられていません。１以上ならkはその番号の覆面記号に割り

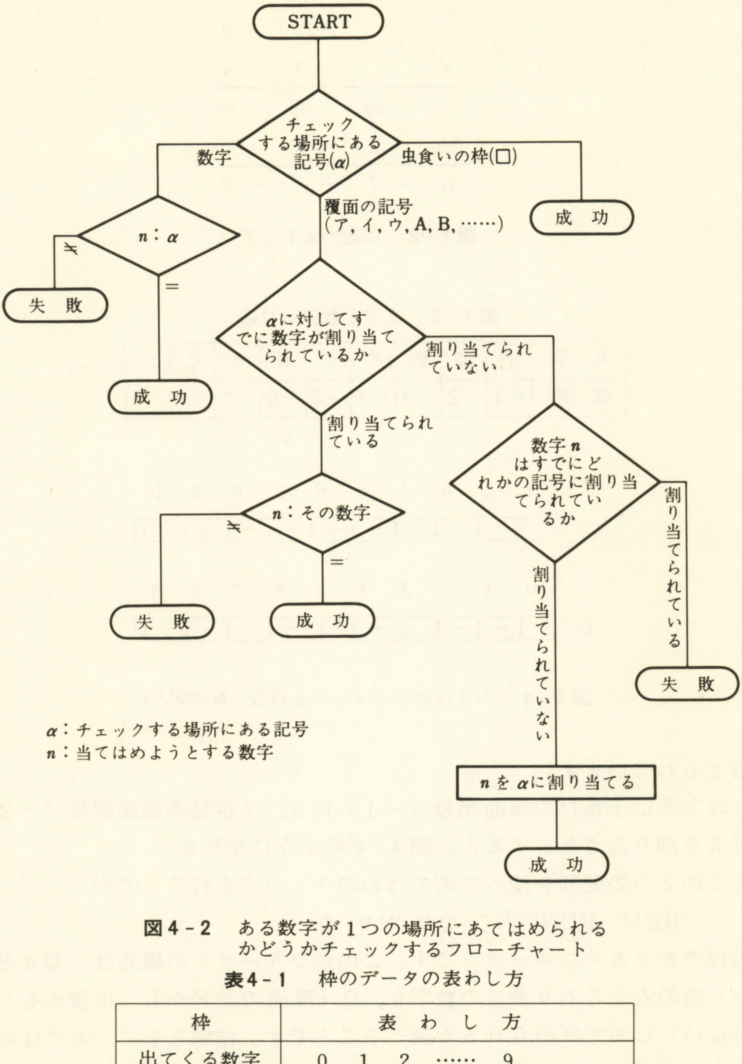


図4-2　ある数字が1つの場所にあてはめられるかどうかチェックするフローチャート

表4-1　枠のデータの表わし方

枠	表わし方
出てくる数字	0，1，2，……，9
覆面の記号	−1，−2，−3，……，−10
虫　　食	99

		−1	−2
×		−3	−4
	99	99	99
99	−3	−5	
−6	−7	−8	−9

図4-3　問題の表わし方

表4-2　文字と数値の対応

文　字	酒	は	飲	め	ま	れ	る	な	よ
数　値	−1	−2	−3	−4	−5	−6	−7	−8	−9

	1	2	3	4	5	6	7	8	9	10
F：	5	3	−1	−1	−1	−1	−1	−1	−1	−1

	0	1	2	3	4	5	6	7	8	9
L：	−1	−1	−1	2	−1	1	−1	−1	−1	−1

図4-4　あてはめのチェックを行なう配列FとL

当てられています。

たとえば1番目の覆面記号（−1）に5，2番目の覆面記号（−2）に3を割り当てたとすると，**図4-4**のようになります。

この2つの配列を使ってあてはめのチェックを行なうのが，

REM　FUNC　C(Q9，N9，Q)

の行で始まるサブルーチンです。このサブルーチンの機能は，Q9という数の右からN9番目の数字を，Q（覆面の記号かも，虫食かもしれない）にあてはめられるか調べることです。結果として，あてはめられるならCを1，あてはめられないならCを0とします。

プログラムを**図4-5**に示します。

```
0010 PRINT "FUKUMEN-MUSHIKUIZAN"
0020 DIM X(10),Y(10),Z(10,11),U(20)
0030 DIM F(10),L(10)
0040 REM 10^(K-1) NO KETA
0050 DEF FNA(W1)=W1-INT(W1/10)*10
0060 DEF FNK(W,K)=FNA(INT(W/10^(K-1)))
0070 READ M,N
0080 FOR M1=1 TO M
0090    READ X(M+1-M1)
0100    NEXT M1
0110 FOR N1=1 TO N
0120    READ Y(N+1-N1)
0130    NEXT N1
0140 FOR N1=1 TO N
0150    FOR M1=1 TO M+1
0160       READ Z(N1,M+2-M1)
0170       NEXT M1
0180    NEXT N1
0190 FOR L=1 TO M+N
0200    READ U(M+N+1-L)
0210    NEXT L
0220 DATA 2,2
0230 DATA       -1,-2
0240 DATA       -3,-4
0250 REM ----X-------
0260 DATA   99,99,99
0270 DATA 99,-3,-5
0280 REM ------------
0290 DATA -6,-7,-8,-9
0300 PRINT "====";M;" KETA X ";N;" KETA"
0310 REM KEISAN
0320 FOR X9=10^(M-1) TO 10^M-1
0330    FOR Y9=10^(N-1) TO 10^N-1
0340       FOR Q=0 TO 10
0350          F(Q)=-1
0360          L(Q)=-1
0370          NEXT Q
0380       FOR M1=1 TO M
0390          Q9=X9
0400          N9=M1
0410          Q=X(M1)
0420          GOSUB 810
0430          IF C=0 THEN 790
0440          NEXT M1
0450       FOR N1=1 TO N
0460          Q9=Y9
0470          N9=N1
0480          Q=Y(N1)
0490          GOSUB 810
0500          IF C=0 THEN 780
0510          Z9=FNK(Y9,N1)*X9
0520          FOR M1=1 TO M+1
0530             Q9=Z9
0540             N9=M1
```

図4-5　覆面算のプログラム ①

```
0550             Q=Z(N1,M1)
0560             GOSUB 810
0570             IF C=0 THEN 780
0580             NEXT M1
0590          IF Z(N1,M+1)<>99 THEN 610
0600          IF FNK(Z9,M+1)=0 THEN 780
0610          NEXT N1
0620       U9=X9*Y9
0630       FOR L=1 TO M+N
0640          Q9=U9
0650          N9=L
0660          Q=U(L)
0670          GOSUB 810
0680          IF C=0 THEN 780
0690          NEXT L
0700       IF U(M+N)<>99 THEN 720
0710       IF FNK(U9,M+N)=0 THEN 780
0720       REM DEKIAGARI
0730       PRINT "KOTAE";X9;" X ";Y9;" / ";
0740       FOR N1=1 TO N
0750          PRINT FNK(Y9,N1)*X9;
0760          NEXT N1
0770       PRINT " / ";U9
0780       NEXT Y9
0790    NEXT X9
0800 GOTO 70
0810 REM FUNC C(Q9,N9,Q)
0820 IF Q=99 THEN 960
0830 K1=FNK(Q9,N9)
0840 IF Q<0 THEN 870
0850 IF K1=Q THEN 960
0860 GOTO 940
0870 IF F(-Q)<0 THEN 900
0880 IF K1=F(-Q) THEN 960
0890 GOTO 940
0900 IF L(K1+1)>=0 THEN 940
0910 L(K1+1)=-Q
0920 F(-Q)=K1
0930 GOTO 960
0940 C=0
0950 RETURN
0960 C=1
0970 RETURN
0980 END
```

図4－5　覆面算のプログラム　②

●サブルーチンの役割

あてはめのチェックのような決まった処理は，普通，1つのサブプログラムにまとめるのがよいのです。あとで述べますが，こうすることによって，プログラムがすっきりした形になり，ミスも少なくなります。

BASICではサブプログラムという考え方はありません。しかし，GOSUB文とRETURN文の組合せによるサブルーチンを，サブプログラムとして使うことがある程度できます。また，非常に簡単な機能ならばFNK (X, Y) のような関数でも実現できます。

このプログラムでは，あてはめのチェックは関数で表わしたいところですが，BASICの関数はこのように複雑なものを表わせません。関数の定義が1行でしか書けないからです。しかたなく，あてはめのチェックはGOSUB文，RETURN文を使って書くことになります。

ところが，GOSUB文には，引数によるデータの受け渡しがありません。引数によるデータの受け渡しとは，

```
DEF  FNF(X)=X*(1-X)
          ⋮
A=FNF(2)*FNF(3)
          ⋮
```

のような場合，Xには2や3が与えられることです。プログラムではしかたなく代入文をGOSUB文の前に置いて，引数の代わりにしています。

```
Q9=X
N9=Y
Q=Z
GOSUB  810
IF  C=0  THEN~
```

というのは，

```
IF  FNC(X, Y, Z)=0  THEN~
```

と書きたいところです。

また，普通，サブプログラムの中では，その中だけで使える変数

表 4－3 ・BASICのサブプログラムの機能

	引数を持つ	2行以上に書ける	局所変数を持つ
FORTRANのサブルーチンや関数などの本来のサブプログラム	○	○	○
BASICの関数	○	×	×
BASICのサブルーチン	×	○	×

（局所変数）がありますが，BASICではこれはもちろんありません。

このようにBASICは，サブプログラムの機能が貧弱です。サブプログラムの機能が貧弱であるということは，大きなプログラムを作るときには**重大な欠点となる**といえます。

BASICのサブプログラム（らしいもの）と本来のサブプログラムの比較を**表 4－3** に書いておきます。

このようにBASICの言語に問題点はありますが，覆面算のプログラムのようにするのがひとつの方法でしょう。次のことが大事です。

決まった処理はサブルーチン
(GOSUB……RETURN) にまとめよう

これをしないと，プログラムが「うなぎの寝床」になってしまいます。サブプログラムの話は大事なので，またあとでもすることにします。

課　題

4―1　次のたし算の覆面算を解いてください。またこのような問題を解くプログラムを作ってください。(解答194ページ)

```
      こ れ
+   あ れ
────────
  あ そ こ
```

図4-6　覆面算

4―2　昔からある覆面算で，次のようなものがあります。解いてみてください。また，これをプログラムで解く問題点はなんでしょうか。(解答194ページ)

```
   SEND           CROSS
+  MORE        +  ROADS
-------        --------
  MONEY          DANGER
```

図4-7　覆面算

4―3　虫食の枠の中にある数（たとえば3）だけは入れてはいけない，というルールはどのようにプログラムしますか。(解答194ページ)

4―4　覆面のところには，奇数だけしか入れてはいけない，というルールがあったら，どうプログラムしますか。(解答194ページ)

休憩室 4

FORループを0回まわる

次のプログラムによってAは何個表示されますか。

```
10  FOR  I=1  TO  K
20      PRINT "A"
30      NEXT  I
40  END
```

AはK個表示されます。**でもK=0だったら？**

多くの計算機では，0個ではなく1個のAを表示してしまうのです。多分計算速度を速くするためでしょうが，これはよくない設計です。自分のパソコンでも調べてみてください。

なぜAを1回表示してしまうかというと，FOR I=1 TO KとNEXT Iの処理が中では，

```
I=1
  :  ←─────────┐
  :            │
I=I+1          │
IF  I<=K  THEN ┘
```

となっているからでしょう。

```
I=1
┌→IF  I>K  THEN ─┐
│    :           │
│ I=I+1          │
└─GOTO           │
     ←───────────┘
```

のように先に終了の判定をするようになっていれば，Aは表示されません。

5★切符の数字

数楽パズル

【問　題】

4つの数字と1つの数を与えます。4つの数字をその出した順に従って並べ，＋，－，×，÷の四則とカッコだけをその間に入れて式をつくり，式の値がその数になるようにしてください。

たとえば，1，9，8，1をこの順で使って式を作り，式の値が8になるようにしてください。

電車の切符には，たいてい4桁の数が付いています。これは切符の通し番号で，0000から9999まで1枚ごとにこの数が増えていきます。電車を待って退屈なときなどこの数字の間に四則（＋，－，×，÷）を入れて式の値が10になるように考えてみたりします。

たとえば，1，9，8，4から10を作るには，

$$-1+9+8\div 4$$

とすればよいわけです。

●切符の数字パズルのルール

このようなパズルにはいろいろなルールがありますが，ここでは次のようなルールのもとでやってみます。もちろんルールが違っていても，ほとんど同じようなプログラムで解くことができます。

① 4個の数字に対して行なう。

② 数字はでたらめに持ってきて問題とする〔4つの同じ数字で行なうものもある（four foursなど）〕。

③ 作る数もいろいろな場合を考える〔10にするというのがよく知られている〕。

④ 数字は並べ換えないでその順で使う〔並べかえを許すというのもある〕。

⑤ 四則（＋，－，×，÷）とカッコを使えることとする。単項の‘－’（－3，－（2＋1）など）も許す〔カッコを許さない，という方法もある。平方根などの他の演算を許す方法もある〕。

⑥ 数字を並べて2桁以上の数として使うのは許さない〔許すとい

うのもある〕。

●切符のパズルで調べる全部の場合の数

4個の数字を組み合わせて式を作るとき，どれだけの場合の数があるでしょうか。プログラムを作る前にこのことを分析しましょう。ここでわかるのは，場合の数を増すもととなるものとして，次の3つがあることです。

① 間に入れる四則の記号の選択
② 計算の順序
③ 途中に単項の‘－’を入れるか入れないか

●間に入れる四則の記号の選択

入れる場所は3箇所あります。おのおの4通りの可能性があります。したがって全部で，

$$4 \times 4 \times 4 = 4^3 = 64\text{通り}$$

の場合があるわけです。

●計算の順序

4個の数を計算する上で，最初の数と次の数との間でなんらかの計算をし，それと3番目の数と計算し，その結果と最後の数と計算するというのもあります。また2番目と3番目とを計算し，それと4番目を計算し，さらに，1番目とその結果とを計算するというのもあるでしょう。よく考えるとこのような計算の順序は，**図5－1**のように5通りあると

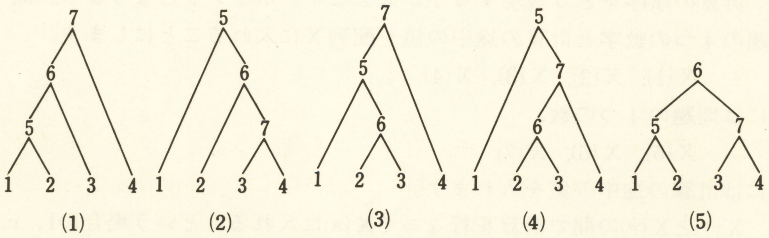


図5－1　計算の順序の種類

いうことがわかります。専門的にいうと2分木（binary tree）で葉の数が4のものの種類は5だ，ということになります。図で5，6，7というのは途中や最後の計算の結果を表わします。たとえば(3)は，

(□•(□•□))•□)

という計算順序ということです（□は数字，•は四則の1つという意味です)。

●単項の‘−’の入れ方

単項の‘−’があると非常に複雑になると思われるかもしれませんが，実は2倍の可能性があるだけなのです。つまり，次のように，単項の‘−’を四則の外側に移すことができ，結局**式の全体に‘−’をつけるかつけないかという2通り**に全部が含まれてしまいます。

$$\left.\begin{matrix} \alpha \left\{\begin{matrix}\times\\ \div\end{matrix}\right\} (-\beta) \\ \\ (-\alpha) \left\{\begin{matrix}\times\\ \div\end{matrix}\right\} \beta \end{matrix}\right\} \longrightarrow -\left(\alpha \left\{\begin{matrix}\times\\ \div\end{matrix}\right\} \beta\right)$$

$$\alpha \left\{\begin{matrix}+\\ -\end{matrix}\right\} (-\beta) \longrightarrow \alpha \left\{\begin{matrix}-\\ +\end{matrix}\right\} \beta$$

$$(-\alpha) \left\{\begin{matrix}+\\ -\end{matrix}\right\} \beta \longrightarrow -\left(\alpha \left\{\begin{matrix}-\\ +\end{matrix}\right\} \beta\right)$$

ここでαやβは数字や式のことで，{ }はそのうち1つを選ぶということです。

●切符のパズル・プログラムの方法

計算の順序をどう表現するかが大きなキーポイントとなります。問題の4つの数字と計算の途中の値を配列Xに入れることにします。

X(1), X(2), X(3), X(4)

には問題の4つの数，

X(5), X(6), X(7)

には計算の途中の値を入れます。

X(l)とX(r)の間で計算を行なってX(a)に入れる，という場合のl，r，aの組をいくつか決めてやれば図5-1のような計算順序が決まります。

初期設定の部分でこのL, R, Aに値が読み込まれます。最初のDATA文のデータを全部読み込むと,

L(1, 1)＝1, R(1, 1)＝2, A(1, 1)＝5
L(1, 2)＝5, R(1, 2)＝3, A(1, 2)＝6
L(1, 3)＝6, R(1, 3)＝4, A(1, 3)＝7

となります。これは,

X(1)とX(2)で計算してX(5)に入れ,
X(5)とX(3)で計算してX(6)に入れ,
X(6)とX(4)で計算してX(7)に入れる

ということ, つまり図5-1の(1)の計算順序で計算するということを表わしています。

計算の順序を答で表わすときには,

1＋(2×(3－4))

のようにカッコによって表わすことにします。図5-1の計算順序に対応して, カッコのパターンを用意しておきます。これには文字列型配列P$を使っています。

文字列型配列の二次元のものはBASICにないので(あるBASICもありますが),

P$(J, M)

の代わりに,

P$(8＊(J－1)＋M)

と書いています(普通の二次元配列では, 8＊(J－1)＋Mのような計算をシステム側で自動的に行なって記憶の場所を参照しているのです)。

配列Sには四則のうちどれを使うかという情報を入れておきます。その値1, 2, 3, 4に＋, －, ×, ÷が対応します。S(1), S(2), S(3)を1から4に, J(計算の順序の方式)を1から5に変えて全部の場合を作ります。

ひとつの場合を決めた後, 作る目標の数Yと計算した結果の数が一致するか調べます。ここでは計算した結果X(A(J, 3))に絶対値をとり, 符号を除いて比較しています。符号が目標Yと違っていても,

前にも述べた単項の‘−’を式に付ければいいわけです。もうひとつ大事なことを次に述べます。

●1÷10×10はなに？

BASICで次の式の値はどうなるでしょうか。

```
1/10*10
```

1になると思ったら，ちょっと甘いですね。普通BASICでは1/10は整数でないので，内部で実数型で表わされるようになります。また数は2進ですから1/10は割り切れません。つまり1/10を正確に数値表現することはできません。実際は0.09999998……のような少しずれた値をとることになります。したがって，1/10*10も正確に1になることはありません。

このパズルでも四則の中に割り算が入っています。ですから，このようなことが起こることがあるわけです（1，9，8，1から8を作る，というのにも割り算が大事な役割を果たしています)。ところがパズルでは，1/10*10が1であるということが判定できなければうまくいきません。ここをうまく行なうために，少し幅を持たせて等しいかどうか判定するという手法を使っています。すなわち，xとyが等しければ100へ行くというのを，

```
IF x=y THEN 100
```

ではなく，

```
IF ABS(x-y)<0.00001 THEN 100
```

とするのです。

次のことを忘れないでください。

> 実数には誤差が常にある。
> 精度は無限ではない。

●切符のパズルのプログラム

プログラムを**図 5 - 2**，結果の一例を**図 5 - 3** に示します。プログラムの性質上むだなカッコは残っています。

プログラムでは解の数を数えて表示するようになっています。

```
010 PRINT "KIPPU PUZZLE: SUURAKU"
020 DIM L(5,3),R(5,3),A(5,3)
030 DIM X(7),P$(40),M$(2),N$(2)
040 DIM S$(4),S(3)
050 REM SHOKISETTEI
060 FOR J=1 TO 5
070   FOR K=1 TO 3
080     READ L(J,K),R(J,K),A(J,K)
090     NEXT K
100   NEXT J
110 DATA 1,2,5,5,3,6,6,4,7
120 DATA 3,4,7,2,7,6,1,6,5
130 DATA 2,3,6,1,6,5,5,4,7
140 DATA 2,3,6,6,4,7,1,7,5
150 DATA 1,2,5,3,4,7,5,7,6
160 FOR S1=1 TO 4
170   READ S$(S1)
180   NEXT S1
190 DATA "+","-","*","/"
200 FOR J=1 TO 5
210   FOR M=1 TO 8
220     READ P$(8*(J-1)+M)
230     NEXT M
240   NEXT J
250 DATA "((","  ","  ",") ","  ",") ","  ","  "
260 DATA "  ","  "," (","  "," (","  ","  ","))"
270 DATA " (","  "," (","  ","  ","))","  ","  "
280 DATA "  ","  ","((","  ","  ",") ","  ",") "
290 DATA " (","  ","  ",") "," (","  ","  ",") "
300 READ M$(1),N$(1),M$(2),N$(2)
310 DATA "-(",") ","  ","  "
320 REM NYUURYOKU
330 T=0
340 PRINT "4 コ ノ カス゛";
350 INPUT X(1),X(2),X(3),X(4)
360 PRINT "ツクル カス゛";
370 INPUT Y
380 REM KEISAN
390 FOR S1=1 TO 4
400   S(1)=S1
410   FOR S2=1 TO 4
420     S(2)=S2
430     FOR S3=1 TO 4
440       S(3)=S3
450       FOR J=1 TO 5
460         FOR K=1 TO 3
470           ON S(A(J,K)-4) GO TO 480,500,520,540
```

```
480             X(A(J,K))=X(L(J,K))+X(R(J,K))
490             GO TO 560
500             X(A(J,K))=X(L(J,K))-X(R(J,K))
510             GO TO 560
520             X(A(J,K))=X(L(J,K))*X(R(J,K))
530             GO TO 560
540             IF X(R(J,K))=0 THEN 720
550             X(A(J,K))=X(L(J,K))/X(R(J,K))
560             NEXT K
570           REM ICCHI NO HANTEI
580           IF ABS(Y-ABS(X(A(J,3))))>1E-6 THEN 720
590           H=2
600           IF Y*X(A(J,3))>0 THEN 620
610           H=1
620           T=T+1
630           REM KOTAE HYOUJI
640           PRINT Y;" = ";M$(H);
650           FOR M2=1 TO 4
660             PRINT P$(8*(J-1)+2*M2-1);X(M2);
670             PRINT P$(8*(J-1)+2*M2);
680             IF M2=4 THEN 710
690             PRINT S$(S(M2));
700             NEXT M2
710           PRINT N$(H)
720           NEXT J
730         NEXT S3
740       NEXT S2
750     NEXT S1
760 PRINT X(1);X(2);X(3);X(4);"==>";Y;
770 PRINT "  KAI NO KAZU";T
780 GO TO 320
790 END
```

図 5-2　切符のパズルのプログラム

問題の答は図のように，

　　1/（9/8 － 1）

でした。途中の計算の値が整数でなくなるところが面白いところです。そのせいで，この問題は非常にむずかしいものになっています。

課　題

5－1　1，9，8，4の数字から式を作って，0から100までの数をおのおの作り出せるか調べてください。作り出せるときはおのおの一例となる式を出力すること。（解答195ページ）

```
KIPPU PUZZLE: SUURAKU
4 コ ノ カズ"?1,9,8,1
ツクル カズ"?8
     8   =          1   /((    9    /        8 ) -        1 )
     1      9       8      1 ==>     8   KAI NO KAZU    1
4 コ ノ カズ"?1,9,8,2
ツクル カズ"?8
     8   =     (    1   -      9 ) + (       8   *        2 )
     8   = -( (     1   - (    9   +         8 ))/        2   )
     8   =          1   - (    9   - (       8   *        2 ))
     8   = -(((     1   -      9 ) -         8 ) /        2   )
     1      9       8       2 ==>    8   KAI NO KAZU    4
4 コ ノ カズ"?4,4,4,4
ツクル カズ"?4
     4   =          4   +((    4   -         4 ) *        4 )
     4   =          4   +((    4   -         4 ) /        4 )
     4   =          4   + (    4   * (       4   -        4 ))
     4   =          4   -((    4   -         4 ) *        4 )
     4   =          4   -((    4   -         4 ) /        4 )
     4   =     ((   4   -        4 ) *       4 ) +        4
     4   = -(((     4   -        4 ) *       4 ) -        4   )
     4   =          4   - (      4   * (     4   -        4 ))
     4   =     ((   4   -        4 ) /       4 ) +        4
     4   = -(((     4   -        4 ) /       4 ) -        4   )
     4   =     (    4   * (      4   -       4 ))+        4
     4   = -( (     4   * (      4   -       4 ))-        4   )
     4      4       4      4 ==>     4   KAI NO KAZU   12
4 コ ノ カズ"?
```

図 5 - 3　切符のパズル結果の例

5—2　1980から2001までの数の各桁の数字から式を作って，10にできるかどうか調べてください。（解答196ページ）

5—3　1926から2025まで，各桁の数字から式を作り，その数を西暦の年と見て，それに対応する昭和の年の数になるかどうか調べてください。

つまり，

1，9，2，6から1を

1，9，2，7から2を

⋮

1，9，8，1から56を

1，9，8，2から57を

⋮

2，0，2，5から100を作ってみてください。(解答196ページ)

5—4　数の並びの間に＋や－などの記号を入れないで，その並びを2桁以上の数と見てもよいことにするには，プログラムをどう改めればよいのでしょうか。つまり1，9，8，1から10を作るのに

19－8－1＝10

としてもよいとした場合です。(解答196ページ)

5—5　4個の数字を並べ換えてもよいとした場合はプログラムをどう変えればよいでしょうか。(解答198ページ)

休憩室 5

1980年（昭和55年）は何の年

この年は珍しい年です。めったにありません。パズル愛好家にはよく知られています。次のプログラムをよく見てください。

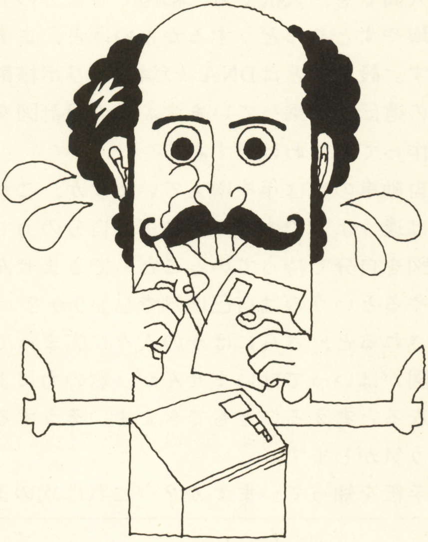
```
0010 FOR S=1 TO 1930
0020    A=S+1925
0030    IF INT(A/S)<>A/S THEN 50
0040    PRINT A;"=";A/S;"*";S
0050    NEXT S
0060 END
```

```
          1926 = 1926 *    1
          1930 =  386 *    5
          1932 =  276 *    7
          1936 =  176 *   11
          1950 =   78 *   25
          1960 =   56 *   35
          1980 =   36 *   55
          2002 =   26 *   77
          2100 =   12 *  175
          2200 =    8 *  275
          2310 =    6 *  385
          3850 =    2 * 1925
```

西暦年を昭和年で割り切る年

6★幸福の手紙

自分を表示するプログラム



●自己増殖と幸福の手紙

自己増殖とはなんでしょうか。生物が増えるのはみな自己増殖といえます。人間でも，大根でも，細菌でも自分の力で数が増えていきます。数を増やすときにどうするかというと，まず設計図を複製（コピー）します。設計図とはDNA（デオキシリボ核酸）の列で，その上にたくさんの遺伝子が乗っています。この設計図を読んで，蛋白質を作り身体を作っていくわけです。

一方，自動車の数は年々増えていますが，これは自己増殖でしょうか　これは違うようですね。自動車は自分の力で増えるわけではないし，設計図を自分で持っていることもできません。

歌がはやるというのは自己増殖でしょうか？　歌が少し歌われ始めてもてはやされると，さらにほかの人々に広まっていきます。歌の中に歌の設計図がはいってはいませんが，歌の力によって多くの人々が歌うようになると考えることもできます。そうすると，少し自己増殖かな，という気がします。

幸福の手紙を知っていますか？　これは次のようなものです。

> **これは幸福の手紙です。一週間以内にこの手紙とまったく同じ手紙を，だれでもいいから30人に出すことを勧めます。**
>
> **そうすればあなたには幸福が約束されます。もし手紙を出さなければ，あなたは一年以内に不慮の事故に遭うでしょう。**

こういうものを出してはいけませんが，科学の立場から見ると面白いものです。

幸福の手紙が広まっていくのは自己増殖でしょうか？まず幸福の手紙は，その手紙の力で伝わっていくといってよいでしょう。流行歌がはやるというのは，歌手やレコード会社がはやらせようとしたからはやる，という面もあります。幸福の手紙の場合は，ある程度広まればどんどんさらに広まっていきます。こうなれば特定の人が広めよう

としたために広まるのではありません。たいていの人は書きたくないいが気持が悪いので，いやいやながら書くのです。つまり手紙の力で手紙が増えるということになりましょう。

次に幸福の手紙の中心部分を取り出してみると，

この手紙と同じものを書いて出せ

ということになります。「同じものを書け」といっているのですから，この手紙自身がその設計図であるということができます。

このように考えてくると，.幸福の手紙は，自動車や流行歌に比べて，かなり自己増殖に近い増え方をするということになります。

この幸福の手紙のようなBASICのプログラム，つまり自己増殖するプログラムを作れないか，というのがこの章の問題ですが，その前に設計図（遺伝子）と身体について考えてみましょう。人間でも細菌でも自分の身体そのものはコピーできません。コピーされるのは遺伝子であって，それを読んで身体ができてくるわけです。上の幸福の手紙では身体全体が設計図になっていましたから，生物の自己増殖と少し違う手ぬきの自己増殖といえましょう。そこで次の問題を考えてみましょう。

【問　題】

指令（コピーできない部分）と，設計図（コピーできる部分）とでできた幸福の手紙を作れ。

この答の一例は**図6－1**のようになります。これが後で作るBASICのプログラムに関係してくるのです。自己増殖とは実に奇妙なものですね。

ここで枠の中の部分は，単なる文字の列だと考えてください。つまり枠の中は実行されません。実行されるのは指令の部分だけです。こうなるとかなり生物の自己増殖に近くなってきます。

さて，本題に進みましょう。

まず枠の中の文を枠ごとコピーすること。その下に枠の中の文だけをコピーすること。（枠あり）	コピーする部分
まず枠の中の文を枠ごとコピーすること。その下に枠の中の文だけコピーすること。	指令

図6－1　指令とコピーする部分が分れた幸福の手紙

●自分を表示するBASICのプログラム

BASICのプログラムで自己増殖をするものを作りたいわけですが，次のことがあるために自己増殖，つまり同一の新しいプログラムを生み出して動作させるということができません。

すなわち，BASICの中で扱っている文字列をプログラムとしてメモリにロードすることはできません。もちろんそこへ制御を移すこともできません。もし文字列をプログラムの一部としてメモリにロードする機能を持つものとしてCALL　LOAD（"……"）があったとすると，

```
10  CALL  LOAD（"20  PRINT  123"）
30  END
```

を実行するとプログラムは，

```
10  CALL  LOAD（"20  PRINT  123"）
20  PRINT  123
30  END
```

となり，123と印字することになるわけですが，現在のところ，こんなBASICは考えられていないようです。LISPという言語にはこのような柔軟性があります。

こういうわけで自己増殖はできないので，自分自身を表示する，ということを考えてみます。ディスプレーに表示できれば，PRINT文をPRINT＃文に置き換えることによって，自分自身をカセットやフロッ

ピーディスク上のファイルに書き出すことができます。これをメモリにロードするという作業を人間が手伝ってやれば，自己増殖的にはなります。さて，問題を明確に述べてみましょう。

【問　題】

LISTコマンドで表示したものと，RUNコマンドによって動作した結果の出力とが同じプログラムはどんなものか？

少しずるい答としては，次のものがあります。

```
10 LIST
20 ┐
30 ├ なんでもよい
 : ┘
```

たいていのBASICでは，LISTなどのコマンドを行番号の後に書くことができます。そしてそのプログラムの動作時に，そのコマンドを実行します。ただし，コマンドを実行した後は，次の行には制御が移らず，制御がBASICのモニタのレベルにもどるのが普通です。したがって上のプログラムをRUNさせると，プログラム全体をリストした後，停止します。

このプログラムは，LISTというBASICでないものを使っています。自分自身を設計図としてコピーしているわけですから，前にも述べたようにあまり面白くはありません。

ではどのようにプログラムを作ればよいのでしょうか？　BASICのプログラムは，自分自身をデータとして読むことはできません(LISPならできます)。したがって，必然的に指令とは別に設計図（コピーするもの）が必要になってきます。

設計図をDATA文で表わしてみます。DATA文を置く位置ですが，いろいろ考えるとプログラムの最初に置くのがよいのです。さらに行番号を付けることなどを考えると，**図6-2**のようになりました。うまくできているでしょう。図6-1ともきれいに対応していることがわかります。

```
 10 DATA FOR N=10 TO 100 STEP 10
 20 DATA READ A$
 30 DATA PRINT N;"DATA ";A$
 40 DATA NEXT N
 50 DATA RESTORE
 60 DATA FOR N=110 TO 200 STEP 10
 70 DATA READ A$
 80 DATA PRINT N;A$
 90 DATA NEXT N
100 DATA END
110 FOR N=10 TO 100 STEP 10
120 READ A$
130 PRINT N;"DATA ";A$
140 NEXT N
150 RESTORE
160 FOR N=110 TO 200 STEP 10
170 READ A$
180 PRINT N;A$
190 NEXT N
200 END
```

図6－2 自分自身を表示するプログラム

課　題

6－1　図6－2の行番号30の行を,

30 PRINT N ; "DATA" ; "PRINT 123"

と書き換えてみます。このプログラムを動作させた結果，出力されたものをプログラムとして入力し，さらにそれを動作させます。その結果を入力し，……というように繰り返していくと何回目でプログラムは動かなくなりますか。

フロッピーディスクがあればPRINT文を全部PRINT＃文になおしてファイルに結果を出力し，さらにそれをロードするようにするとよいでしょう。(解答198ページ)

6－2　図6－2のプログラムを，工夫してもっと短くしてください。(解答198ページ)

6－3　動作させると，自分自身の各行を逆順で表示するようなプログラムを作ってください。

ヒント：配列を使うとやりやすいでしょう。(解答198ページ)

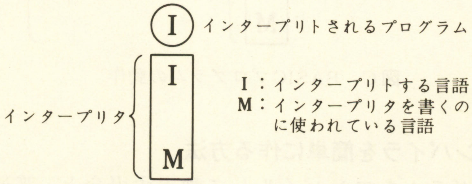
Tの話：コンパイラとインタープリタ

BASICやFORTRAN，PASCALといった計算機言語で書いたプログラムは，どのようにして動作するのでしょうか？　コンピュータはマイコンから大型機まで，無条件で動くプログラムは機械語で書いたプログラムだけです。BASICなどの言語で書いたプログラムは，いろいろな方法によって動くようになります。方法は大別して2つあります。コンパイラによる方法とインタープリタによる方法です。

BASICのプログラムは，普通インタープリタによって動きます。ある言語のインタープリタは，その言語で書かれたプログラムを見ながらそれを**解釈し**，その意味の通りの動作をします。インタープリタ自身は機械語で書かれた1つのプログラムです。

もう1つの方法はコンパイラを使います。FORTRANのプログラムはこの方法で処理されます。ある言語のコンパイラは，その言語で書かれた言語のプログラムを見てそれを機械語のプログラムに**翻訳し**ます。動作するのは，この機械語のプログラムです。

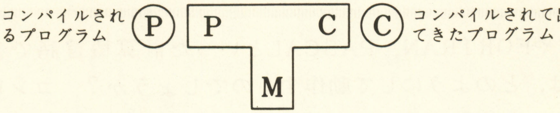
インタープリタを図に書くときはI型にします。その中の上の方にインタープリトする言語の名前，その中の下の方にインタープリタを書くのに使われている言語（普通は機械語）を書きます。インタープリトされるプログラムをインタープリタの上に書いて，全体が動作することを表わします（**図A**）。



図A　インタープリタ

コンパイラを図に書くときは凸を逆さにした型に書きます。左上にコンパイルされる言語の名前，右上にコンパイルされた結果の言語（機

機語が多い）の名前，中央の下にはコンパイラを書くのに使われている言語（機械語が多い）を書きます。言語Pで書かれた言語をコンパイルして，機械語Cにするのを**図B**のように書きます。

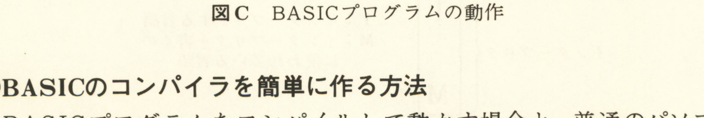


P：コンパイルする言語
C：コンパイルされた結果出る言語(機械語など)
M：コンパイラを書くのに使われている言語(機械語など)

図B　コンパイラ

●BASICプログラムの動き方

BASICは普通インタープリタで動きますが，処理を単純にし計算スピードを上げるために，中間コードに変換してからインタープリトしています。したがって，コンパイルというほどではありませんが，BASICのプログラムを中間コードに翻訳する段階があります。BASIC，中間コード，機械語を，おのおのB，C，Mとすると，プログラムの動き方は**図C**のようになります。BASICの中間コードは，メモリ・マップに書かれてあるBASICのプログラム領域をPEEKしてのぞいて見ればわかります。



図C　BASICプログラムの動作

●BASICのコンパイラを簡単に作る方法

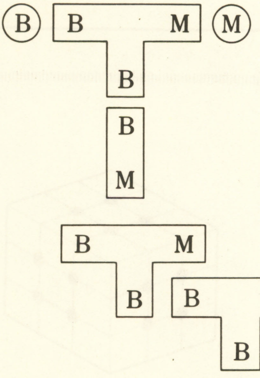
BASICプログラムをコンパイルして動かす場合と，普通のパソコンのように，インタープリタでプログラムが動く場合などの計算時間を比べてみると，**表A**のようになります。直接機械語（あるいはアセン

表A　計算時間を比べる

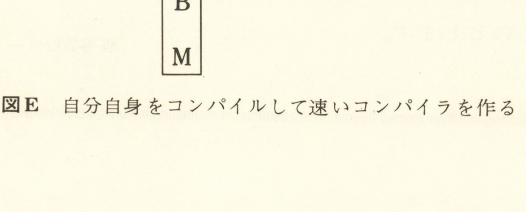
直接機械語で書いた場合	1	数倍
コンパイルした場合	約5	数倍
インタープリタで動く場合	約30	

ブラ）で書いたものにはかないませんが，できればコンパイルした方が速く，望ましいわけです。

コンパイラを作るのは簡単ではなく，何人ものプロが数か月にわたって作られるものです。ここでは，少し簡単な方法（といっても大変ですが）を述べます。今，BASICのインタープリタ $\begin{array}{|c|}\hline B\\M\\\hline\end{array}$ があるとします(中間コードは面倒なので考えないことにします)。コンパイラを機械語で書くのは大変なのでBASICで書いてしまうのです。こうすれば**図D**のように，なんとかコンパイル作業はできます。しかし，これではコンパイル作業が遅すぎるので，**図E**のように自分自身をコンパイルして機械語でできた速いコンパイラを作ります。その後は，これでコンパイルすればよいのです。



図D　インタープリタで動くBASICで書かれたコンパイラ



図E　自分自身をコンパイルして速いコンパイラを作る

●コンパイラのパズル

さて，Tがたくさんからみあう問題を出します。ゆっくり考えてください。

【問　題】

プログラミング言語Pのプログラムは，中間コードCにコンパイルされ，さらにCで書かれたものが機械語のプログラムに翻訳され，それが実際に動作するものとします。今，コンピュータ1とコンピュータ2があり，その機械語を，おのおのM_1, M_2とします。そこにあるものは，中間コードCを機械語M_1に変換する機械語M_1で書かれたプログラム[C M_1 / M_1]，PをCに変換するCで書かれたプログラム[P C / C]，CをM_2に変換するPで書かれたプログラム[C M_2 / P]だけです。

これだけからコンピュータM_2の上だけで，Pで書かれたプログラムをコンパイルして動くようにしてください。

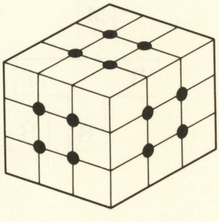
ヒント： [P C / M_2]，[C M_2 / M_2]を作る。

（解答210ページ）

休憩室 6

まっすぐ進めない蟻

図のような立方体の上の線の上を通り，全部の黒い点を1回ずつ回り，元の場所にもどってきてください。ただし，ある黒点から出発し，また黒点を通りぬけるときは，1回しかまっすぐには行けないものとします。



（解答209ページ）

7★桂馬道の問題

【問　題】

将棋盤のような碁盤目の盤を考えます。その上をチェスのナイト（騎士）の動きをする駒で，ひとつのマス目を1回だけ通って全部のマス目をまわる道をみつけてください。

問題を詳しく説明してみましょう。まずナイトという駒の動きを見てみます。これは**図7－1**のようになります。将棋の桂馬に似ていますが，桂馬のように前へ進むだけではなく，8つの方向へ進めます。

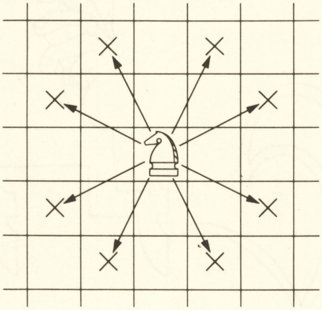


図7－1
ナイトの進み方

このナイトという駒で，正方形や長方形の盤のマスを全部まわるのです。同じところを2度通ってはいけません。この道を桂馬道といいます。出発点はどこから始めてもかまいません。

では簡単な場合を調べてみましょう。縦が3，横が4（3×4）の長方形の盤ならどうなるでしょうか。このときは，**図7－2**のような桂馬道があります。

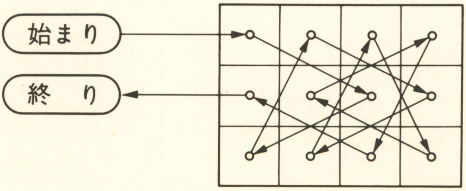


図7－2　3×4の桂馬道

●人間の解く問題

プログラムを作る前に考えてみましょう。4×4の盤，5×5の盤の桂馬道を見つけてください。できますか？「自分はどう考えて道を見つけているのか」ということを考えてください。ここが大事なところです。できなくてもいいのです。

●桂馬道プログラム

桂馬道プログラム全体のリストは図7-10，その動作の結果は図7-11にあります。このプログラムをどのようにして作るのか述べてみましょう。まずデータを表わす方法を考えます。さらにプログラムを小さな部分に分けながら設計してみます。

●盤の表わし方

やはり形の通り，二次元の配列で表わすのがよいでしょう。B(X，Y）によって盤の上からX番目，左からY番目のマスを示すことにします。B(X，Y）には，まだ通っていないときには0，そこを通ったときには，最初から何番目に通ったかという番号を入れておくことにします。こうすれば，配列Bをそのまま出力すれば経路がわかります。たとえば**図7-3**のように進んだとすると，Bは**図7-4**のようになることになります。

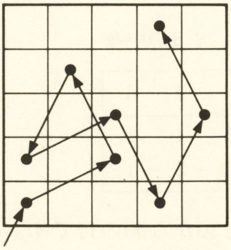


図7-3　ナイトが途中まで進んだ図

X＼Y	1	2	3	4	5	
1	0	0	0	8	0	
2	0	3	0	0	0	
3	0	0	5	0	⑦	B(3,5)＝7
4	4	0	2	0	0	
5	1	0	0	6	0	

図7-4　図7-3の状態のときの配列Bの中身

●ナイトの動き方を表わす

ナイトの動き方（図７－１）を表わすには，**図７－５**のように元いた場所からのずれを使うのがよいでしょう。ずれの値は**表７－１**のように，Ｘ８とＹ８という配列に入れておきます。これを使うと，今（Ｘ，Ｙ）という場所にいてＤ１の方向に進むとすれば（Ｄ１は１から８の数），

Ｘ＝Ｘ＋Ｘ８（Ｄ１）

Ｙ＝Ｙ＋Ｙ８（Ｄ１）

とすればよいことになります。

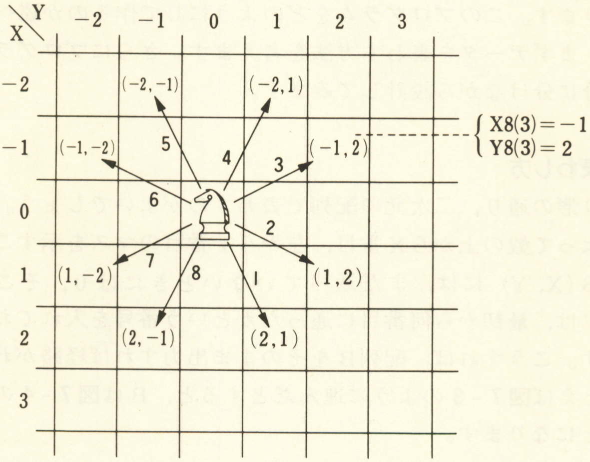


図７－５　ナイトの動き方を表わす位置のずれ

表７－１　ナイトの動き方を表わす配列Ｘ８，Ｙ８の中身

Ｄ１	1	2	3	4	5	6	7	8
Ｘ８（Ｄ１）	2	1	−1	−2	−2	−1	1	2
Ｙ８（Ｄ１）	1	2	2	1	−1	−2	−2	−1

このずれのデータの設定をするのが，行番号30から90の行です。こうすれば，８つの方向を調べるのは，

⋮

ＦＯＲ　Ｄ１＝１　ＴＯ　８

```
X 1 = X + X8 (D 1)
Y 1 = Y + Y8 (D 1)
[D 1方向の処理]
NEXT  D 1
  ⋮
```

とすればいいわけです。このように，ナイトの動き方のような**規則的に表わせる処理，似たものの繰返しの処理はデータで表わすのがよい**のです。次のようにすることはすぐ考えつきます。

```
    ⋮                  ┐
X 1 = X + 2            │
Y 1 = Y + 1            │
[方向 1 の処理]         │
X 1 = X + 1            │
Y 1 = Y + 2            │
[方向 2 の処理]         ├ 8回繰り返す
X 1 = X − 1            │
Y 1 = Y + 2            │
[方向 3 の処理]         │
    ⋮                  ┘
```

しかし，このように**規則的に表わせる処理，似たものの繰返しの処理をプログラムで表わすのはよくない**のです。ではナイトの方向のようなことがらをデータで表わすと，なぜよいのでしょうか。並べてみると，

(1) データで表わすと，なんといってもプログラムが短くなる（したがって虫が少なくなる）。

(2) プログラムを変えるとき，1箇所だけ変えればよい（したがって虫をとりやすい）。

(3) 同じ処理は，プログラムの中の1つの場所で行なうので，わかりやすいプログラムになる。

(4) データを変えることによって，いろいろなものに応用できる（汎用性がある）。

などいろいろな得があります。プログラムで表わすのが得か，データで表わすのが得か，よく考えてみよう！というわけで，

なるべくデータで表わそう

●途中まで進んで行けなくなったらもどる

図7-3のように，ナイトが途中まで進んでどの方向にも行けなくなったら，別の道があるところまでもどる必要があります。それから新しい道を調べていくわけです。また，この「もどる」というのはどこまでもどるかというのも決っていませんし，新しい道でまたもどるということもあります。

もどり方は，盤を表わす配列Bを見てもできるのですが，まわりの8つの方向を見なければならないので，少し面倒です。そこでスタックというものを使います。

●スタック

スタックは，FILO（first in last out）リストとも呼ばれます。計算機の学問の中でも大事なもののひとつで，いろいろなところに使われます。パソコンのメモリ・マップの中にも，スタック領域というのがあるのを御存知でしょう。

スタックはたくさんのものを格納する1つの領域で，FILOというように，最初に格納したものが最後に出るようになっています。たとえば，けん銃の弾倉や細長い駐車場を思いうかべてください（**図7-6**），最初に入れた弾や車が最後に入れたものは最初に出てきます。

スタックをBASICのプログラムの中で，実現する方法を述べましょう。スタックには多くのものが順に格納されるので，その本体として1つの配列を使います。また，どこまでデータが格納されているかを示す，スタックポインタという変数も必要です。スタック本体の配列

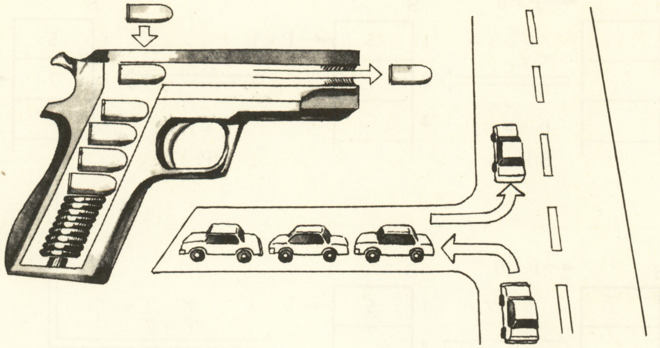


図7－6　スタックのイメージ（弾倉と細長い駐車場）

をS，スタックポインタをPとすると，スタックの初期設定は次のようにします。

```
DIM　S(500)
P＝0
```

スタックにはまだなにもはいっていないので，Pは0とします。

スタックになにかを格納するのをプッシュ・ダウン（push down）といい，Xを格納するとすると，

```
P＝P＋1
S(P)＝X
```

というようにします。

逆に，スタックからデータを取り出すのを，ポップ・アップ（pop up）といい，取り出してXに格納するとすると，

```
X＝S(P)
P＝P－1
```

というようにします。

たとえば，5，7をプッシュ・ダウンしてから，X，Yにポップ・アップしたものを格納すると，**図7－7**のようにスタックは変化します。このように配列の制限はありますが，いくらでも多くのデータを格納し，それを取り出すことができます。

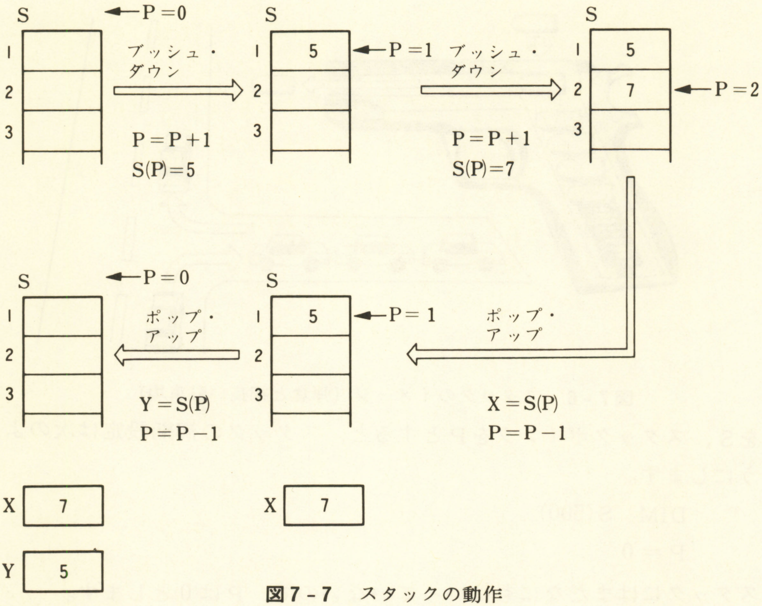


図7－7　スタックの動作

●ナイトが進むサブルーチンともどるサブルーチン

Ｄ１の方向へナイトが１手進むサブルーチンは，次のようになります。

```
REM  SUB  IKU
X＝X＋X８(Ｄ１)
Y＝Y＋Y８(Ｄ１)
N＝N＋１               ：REM  PUSH
Ｄ９(N)＝Ｄ１          ：REM  DOWN
B(X, Y)＝N
RETURN
```

ここで，ナイトの進む方向を格納しておくスタックとして，Ｄ９を使っています。Nはそのスタックポインタですが，ナイトの進んだ手数も表わします。進むだけならスタックはいらないのですが，あとで行

けなくなってもどるときのために，どっちの方向に行ったかということを，スタックにプッシュ・ダウンして覚えておくわけです。

ナイトが1手もどるサブルーチンのすることは，ひとつにはスタックをポップ・アップして，直前にどっちの方向に行ったかということを取り出し，それを見て1手前の場所にもどることです。また，盤に書き込んだ数を消します。そしてプログラムは，

```
REM  SUB  MODORU
B(X, Y) = 0
D1=D9(N)          :REM  POP
N=N-1             :REM  UP
X=X-X8(D1)
Y=Y-Y8(D1)
RETURN
```

となります。このサブルーチンを実行すると，1手進むサブルーチンで変化したもの（現在の位置X，Y，進む方向D1，盤B）が，すべて元にもどります。しかももどるサブルーチンをもう一度実行すれば，さらにもどります。

もどるサブルーチンを繰り返して使えば，何回でも元へさかのぼっていくことができます。

●ナイトが行けるかどうか調べるサブルーチンなど

ナイトが，（X，Y）という位置からD1の方向へ進めるかどうか調べるサブルーチンを作ってみましょう。それには進んだ先（X1，Y1）が，まず(i)盤の外でないこと，それから(ii)前に通ったところでないこと（通ったのなら，そこには1以上の数がはいっている）を調べればよいのです。結局，行番号560〜660のようになります。進めるのならFの値を1，進めないのならFの値を0にするようにしてあります。

●サブルーチンのまとめ

以上のほかに「桂馬道が完成したかどうか」を調べるサブルーチンと，盤の状態（どこまで進んでいるか）を表示するサブルーチンがあ

ります。

これでサブルーチンが合計5個できました。すなわち，それぞれ

① ナイトがD1方向へ1つ進む

② ナイトが1つ前の場所へもどる

③ ナイトがD1方向へ行けるかどうか調べる

④ 現在の状態で桂馬道が完成しているかどうか調べる

⑤ 現在の状態を表示する

という処理を行ないます。これらを材料にして，いよいよ本題を解いてみましょう。

●桂馬道プログラムの中心部分

「ある状態から先へ桂馬道を調べる」というサブルーチンがあるとしてみましょう。すると，主ルーチンは，いろいろな初期設定，特に盤の状態の初期化をしたのち，このサブルーチンを呼べばできあがるわけです。盤の状態の初期化では，配列Bをクリアし，桂馬道を始める地点（X，Y）を決めています。

ではここで最も大切な「ある状態から先へ桂馬道を調べる」サブルーチンを作ってみましょう。まずこの「ある状態」が，どんな状態かで，することが2つに分かれます。

(1) 第一に状態が終りの状態，つまり桂馬道が完成している状態の場合：このときはそのときの盤を表示し「でき上がり」のメッセージを出します。それからプログラムを終了させるか，またはもどり（リターン）ます（プログラムを終了させたときは，桂馬道は1つだけ表示されます。リターンした場合は，可能な桂馬道をすべて表示することになります)。

(2) 第二に終りでない（途中の）状態の場合：このときは8方向に調べを始めます。おのおのの方向に対して次のことを行ないます。まず行けるかどうか調べます。行けないのなら，なにもしません。行けるならば，その方向へナイトを進めます（つまり状態が変わります)。そこで自分自身，つまり「ある状態から先へ桂馬道を調べる」ことを実行します。これが終ったらナイトをもどします

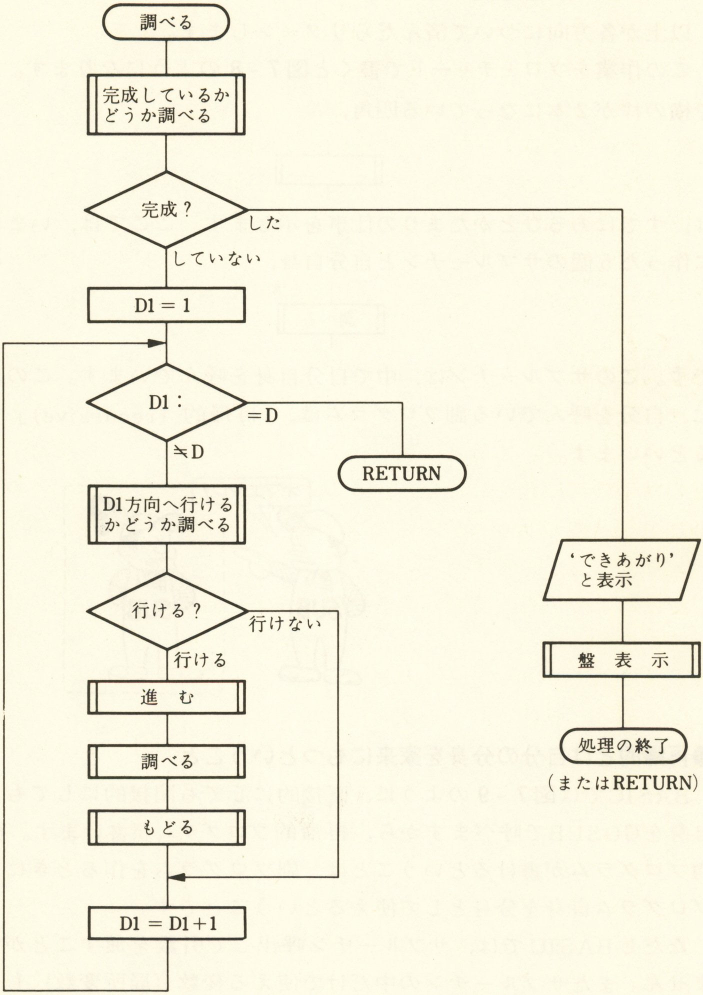
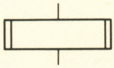


図7-8　「ある状態から先へ桂馬道を調べる」サブルーチン

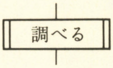
(つまり状態を元にもどします)。

以上が各方向について済んだらリターンします。

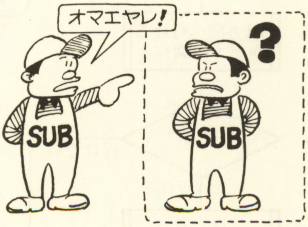
この作業をフローチャートで書くと**図7-8**のようになります。ここで横の枠が2本になっている四角,



は,すでにあるひとかたまりの仕事を示します。ここでは,いままでに作った5個のサブルーチンと自分自身,



です。このサブルーチンは,中で自分自身を呼んでいます。このように,自分を呼んでいる副プログラムは,「再帰的 (recursive)」であるといいます。



●再帰的とは自分の分身を家来にもつということ

BASICでは**図7-9**のように,直接的にしても間接的にしても自分自身をGOSUBで呼べますから,再帰的プログラムが書けます。再帰的プログラムが書けるということは,副プログラムを作るときにそのプログラム自身を分身として使えるということです。

ただしBASICでは,サブルーチン呼出しで引数を渡すことができません。またサブルーチンの中だけで使える変数 (局所変数) もありません。他の計算機言語のLISP,PASCAL,PL/I,Cなどではこれらがすべて可能であり,再帰的プログラムを自由にあやつることができます。BASICはこの点でだいぶ不利ですが,FORTRANのように再帰的プログラムを書けないよりはましです。

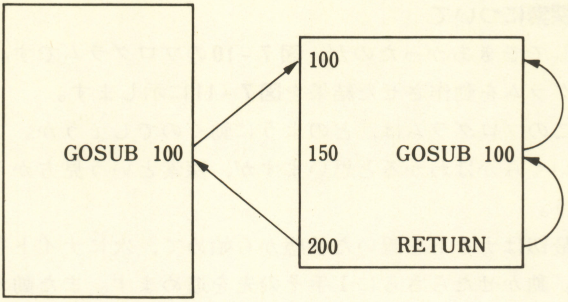


図7-9　再帰呼出し

再帰的プログラムを作ったことのない人は，なじみにくいと思います。しかし食わず嫌いにならないでください。LISPを最初に習った学生は，再帰的でないプログラムを書く方がつらいといいます。そこでひとこと，

> 再帰的感覚を身につけよう！

●縦型探索について

こうしてできあがったのが，**図7-10**のプログラムです。また，このプログラムを動作させた結果を**図7-11**に示します。

ではこのプログラムは，どのように動くのでしょうか。フローチャートでいくらかはわかると思いますが，探索という見方から考えてみましょう。

まず最初はナイトを置いた状態から始めて，次にナイトを1手動かします。動かせたらさらに1手その先を進めます。また動かせたらまた先に進む，というぐあいに進めるうちは，どんどん進んでいきます。だめなら1つ前にもどります。このようにして，全部の可能性を尽していく方法を，縦型探索（depth first search）といいます（**図7-12**）。プログラムは，このやり方で動いているのです。再帰的な呼び出しによって1手先の処理を表わし，うまく縦型探索を実現しています。

一方，1手進んだ状態を全部調べ，2手進んだ状態を全部調べ，3手進んだ状態を全部調べ，……とする方法を横型探索（breadth first search）といいます。これは記憶領域を多量に必要とするため，この問題には適していません（**図7-13**）。

1手2手というように，何手かの組合せで解けるようなパズルは，この縦型探索を使うのが普通です。次の章の「目ひろい」，「詰碁」などがそうですし，ペントミノなどの箱詰めパズル，15ゲームや箱入娘などのスライドブロックパズルも縦型探索で解きます。桂馬道プログラムをよく理解して，いろいろなパズルにチャレンジしてください。

```
0010 PRINT "<ｹｲﾏﾄﾞｳ>"
0020 DIM D9(100),B(8,8),X8(8),Y8(8)
0030 READ D
0040 FOR D1=1 TO D
0050    READ X8(D1),Y8(D1)
0060    NEXT D1
0070 DATA 8
0080 DATA 2,1,1,2,-1,2,-2,1
0090 DATA -2,-1,-1,-2,1,-2,2,-1
0100 PRINT " BAN NO OOKISA";
0110 INPUT X3,Y3
0120 FOR X4=1 TO INT((X3+1)/2)
0130    FOR Y4=1 TO INT((Y3+1)/2)
0140       FOR X=1 TO X3
0150          FOR Y=1 TO Y3
0160             B(X,Y)=0
0170             NEXT Y
0180          NEXT X
0190       PRINT "START NO ICHI";
0200       PRINT X4;Y4
0210       X=X4
0220       Y=Y4
0230       B(X,Y)=1
0240       N=1
0250       GOSUB 300
0260       PRINT "--------------------------------"
0270       NEXT Y4
0280    NEXT X4
0290 STOP
0300 REMSUB SHIRABERU
0310 GOSUB 670
0320 IF F=0 THEN 340
0330 GOSUB 720
0340 FOR D1=1 TO D
0350    GOSUB 560
0360    IF F=0 THEN 400
0370    GOSUB 420
0380    GOSUB 300
0390    GOSUB 490
0400    NEXT D1
0410 RETURN
0420 REMSUB SUSUMU
0430 N=N+1
0440 D9(N)=D1
0450 X=X+X8(D1)
0460 Y=Y+Y8(D1)
0470 B(X,Y)=N
0480 RETURN
0490 REMSUB MODORU
0500 B(X,Y)=0
0510 D1=D9(N)
0520 X=X-X8(D1)
0530 Y=Y-Y8(D1)
0540 N=N-1
```

図7-10 桂馬道プログラム ①

```
0550 RETURN
0560 REMSUB IKERUKA
0570 X1=X+X8(D1)
0580 Y1=Y+Y8(D1)
0590 F=0
0600 IF X1<=0 THEN 660
0610 IF X1>X3 THEN 660
0620 IF Y1<=0 THEN 660
0630 IF Y1>Y3 THEN 660
0640 IF B(X1,Y1)<>0 THEN 660
0650 F=1
0660 RETURN
0670 REMSUB OWARIKA
0680 F=0
0690 IF N<X3*Y3 THEN 710
0700 F=1
0710 RETURN
0720 REMSUB PRINT
0730 PRINT "KOTAE:"
0740 FOR X2=1 TO X3
0750     FOR Y2=1 TO Y3
0760         PRINT B(X2,Y2);
0770         NEXT Y2
0780     PRINT
0790     NEXT X2
0800 RETURN
0810 END
```

図7-10 桂馬道プログラム ②

```
<ｹｲﾏﾄﾞｳ>
 BAN NO OOKISA?3,4
START NO ICHI     1     1
KOTAE:
    1     4     7    10
   12     9     2     5
    3     6    11     8
KOTAE:
    1     4     7    10
    8    11     2     5
    3     6     9    12
------------------------------
START NO ICHI     1     2
------------------------------
START NO ICHI     2     1
KOTAE:
   12     9     6     3
    1     4    11     8
   10     7     2     5
KOTAE:
    8    11     6     3
    1     4     9    12
   10     7     2     5
KOTAE:
   10     7     2     5
    1     4    11     8
   12     9     6     3
KOTAE:
   10     7     2     5
    1     4     9    12
    8    11     6     3
------------------------------
START NO ICHI     2     2
------------------------------
```

図7-11 桂馬道プログラムの動作結果

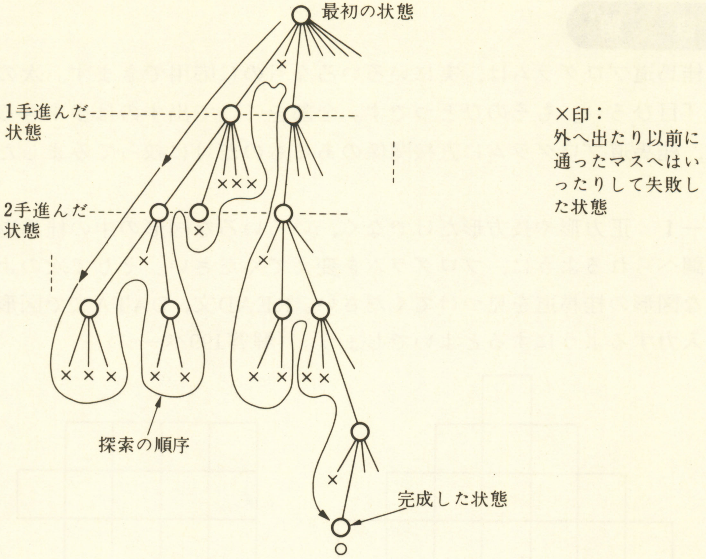


図7－12　桂馬の旅の探索木と縦型探索

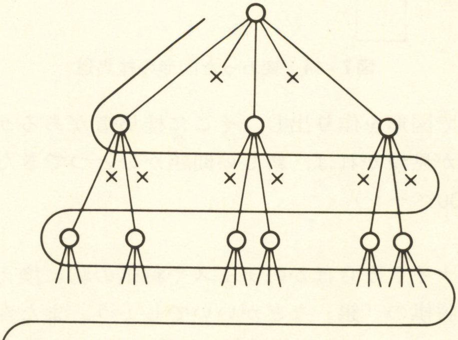


図7－13　横型探索の探索順序

課　題

桂馬道プログラムは，実にいろいろなものに応用できます。次の章の「目ひろい」もそのひとつです。全部をここで出すのは多すぎるので，桂馬道プログラムに直接関係のあるものだけに絞ってみました。

7—1　正方形や長方形だけでなく，いろいろな図形の中の桂馬道も調べられるように，プログラムを変えてください。そして次のような図形の桂馬道を見つけてください。READ文，DATA文で図形を入力するようにするとよいでしょう。(解答199ページ)

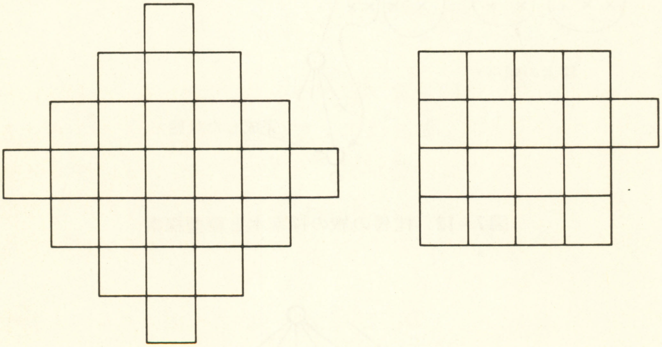


図7-14　変わった図形の桂馬道

7—2　乱数で図形を作り出し，そこに桂馬道があるか調べてください。もし解が見つかれば，新しい問題がひとつできたことになります。(解答200ページ)

7—3　駒をナイトからほかのチェスや将棋の駒に換えてやってみてください。将棋の「銀」などがいいでしょう。また変な効(き)きの駒を自分で作ってやってみるのも面白いと思います(**図7-15**)。さらに駒を換えて7—1，7—2の問題をやってみましょう。(解答200ページ)

7—4　ナイトが全部のマスをまわり，最後に出発点にもどってこな

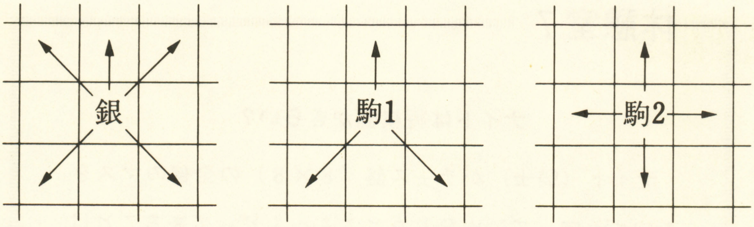


図7－15　違う動きの駒

ければいけない，という問題もあります。閉桂馬道問題とか，ナイトの世界一周問題といいます。これを解くにはプログラムをどう書き換えればよいでしょうか。計算のスピードを考えなければ簡単な方法があります。(解答200ページ)

7－5　この章で作った桂馬道プログラムは，「できる」ことだけを考えたもので，「速くする」ことは考えていません。ですから6×6の盤ぐらいになるとなかなか答が出てきません。8×8のチェス盤ならほとんど不可能でしょう。これも虫食算のところで説明した組合せ論的爆発のひとつです。そこで，プログラムの計算スピードを速くする工夫を考えてください。

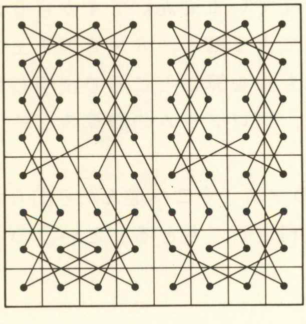
ひとつの方法として，「袋小路のマス（そこへ進んだら出られなくなるマス）ができたら失敗したと判定してもどる」というのが考えられます。(解答200ページ)

休憩室 7

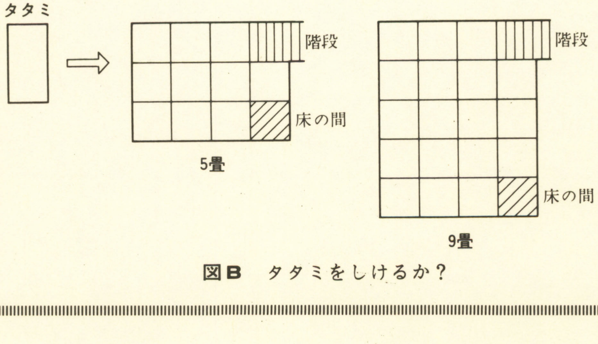
ナイトは将棋盤がきらい?

ナイト（騎士）がチェス盤（8×8）の全部のマスを1回ずつ回って，出発したところへもどって来ることはできます（図A）。ところがナイトは，将棋盤（9×9）にはなじめないらしくて出発したところへもどって来られないようです。ほんとうでしょうか？。

ヒント：次の図のような部屋にタタミをしけますか？（図B）。（解答209ページ）



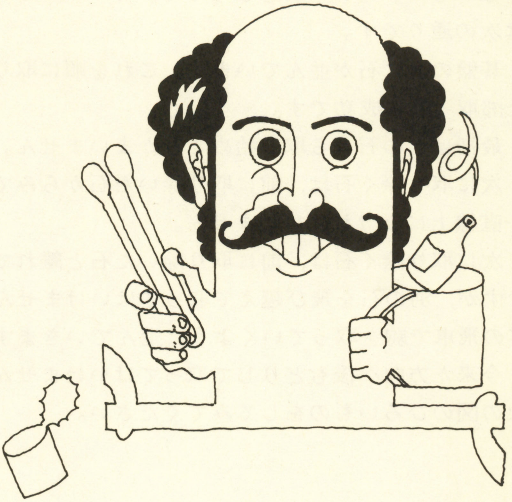
←図A　チェス盤での閉じた桂馬道（閉じたというのは元へもどってくるという意味）



図B　タタミをしけるか？

8★碁石パズル

「ひろいもの」



【問　題】

「ひろいもの」という石をひろっていくパズルがあります。ルールは次の通りです。

1．碁盤の上に石が並んでいます。これを順に取り除いていき，全部取ったら成功です。

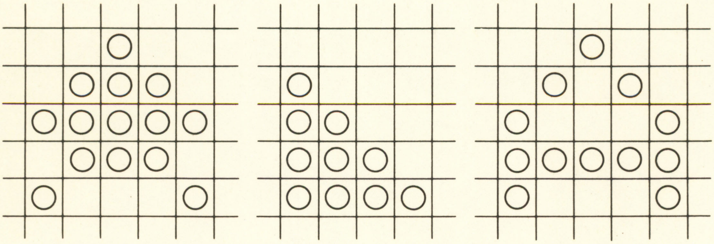
2．最初はどの石から取り始めてもかまいません。

3．次に取り除く石は，前に取り除いた石からみて，縦か横の一直線上になければいけません。

4．次に取り除く石は，前に取り除いた石と離れていてもいいですが，別の石を飛び越えて取ってはいけません（つまり将棋の飛車で駒を取っていくように進んでいきます)。

5．今来た方向へ後もどりして取ってはいけません。

次の図のひろいものをしてみてください。



(a)　ロケット　　(b)　三　角　　(c)　A

図 8－1　ひろいもの

ひろいものというパズルは，「総とり」とか「とらえもの」とも呼ばれ，江戸時代頃からあったもののようです。

解き方の例として，例題の(a)ロケットを考えてみましょう。答の一例は**図 8－2**のようになります。①，②，③，……の順にたどるのが1

つの答です。もちろんこれ以外にもいろいろなたどり方がありますが，少し間違うとすぐに行き詰ります。

まず①のところを見てください。①は上下左右の四方向のうち，下の方向にしか石がありません。ですから①はかならず最初か最後にひろう石になります。まず①をひろってみましょう。①の次は②です。②の次は，③か⑬か⑭がひろえます。しかしここで⑬をひろったりするとうまくいかないのです。よく考えてみてください。①，②，③，④，⑤，……または①，②，③，⑨，⑧，……のようにひろうしかないのです。

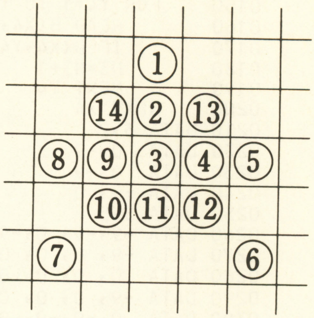


図 8-2　ひろいもの(a)の答の例

●ひろいものプログラムを作る考え方

ひろいものを解くプログラムとその動作結果は，**図 8-3**と**図 8-4**です。このプログラムをよく見てください。前の章の桂馬道を見つけるプログラムによく似ているのがわかると思います。桂馬道とひろいものはかなり違う問題に感じますが，いろいろな共通点が多いのです。

つまり，

① 平面の盤を使い，

② 全部をたどる（あるいはひろう）と成功であり，

③ 動く方向が決まっている

ということは両方とも同じです。

```
0010 PRINT "==ゴイシ パズル:  ヒロイモノ=="
0020 DIM D9(100),B(8,8),X8(8),Y8(8)
0030 READ D
0040 FOR D1=1 TO D
0050    READ X8(D1),Y8(D1)
0060    NEXT D1
0070 DATA 4
0080 DATA 0,1,1,0,0,-1,-1,0
0090 REM NEW PROBLEM
```

図 8-3　ひろいものを解くプログラム　①

```
0100 PRINT "ﾓﾝﾀﾞｲ:"
0110 S=0
0120 N3=0
0130 READ X3,Y3
0140 FOR X4=1 TO X3
0150    FOR Y4=1 TO Y3
0160       READ B(X4,Y4)
0170       IF B(X4,Y4)=0 THEN 190
0180       N3=N3+1
0190       NEXT Y4
0200    NEXT X4
0210 DATA 3,2
0220 DATA  0,-9
0230 DATA  0,-9
0240 DATA -9, 0
0250 DATA 5,5
0260 DATA -9,-9, 0,-9,-9
0270 DATA -9, 0, 0, 0,-9
0280 DATA  0, 0, 0, 0, 0
0290 DATA -9, 0, 0, 0,-9
0300 DATA  0,-9,-9,-9, 0
0310 DATA 4,4
0320 DATA  0,-9,-9,-9
0330 DATA  0, 0,-9,-9
0340 DATA  0, 0, 0,-9
0350 DATA  0, 0, 0, 0
0360 DATA 6,5
0370 DATA -9,-9, 0,-9,-9
0380 DATA -9, 0,-9, 0,-9
0390 DATA  0,-9,-9,-9, 0
0400 DATA  0,-9,-9,-9, 0
0410 DATA  0, 0, 0, 0, 0
0420 DATA  0,-9,-9,-9, 0
0430 GOSUB 1160
0440 FOR X=1 TO X3
0450    FOR Y=1 TO Y3
0460       IF S=1 THEN 570
0470       IF B(X,Y)<>0 THEN 540
0480       REM INIT
0490       B(X,Y)=1
0500       N=1
0510       D9(1)=-99
0520       GOSUB 600
0530       B(X,Y)=0
0540       NEXT Y
0550    NEXT X
0560 PRINT "ｺﾚ ﾊ ﾃﾞｷﾅｲ"
0570 PRINT "=================================="
0580 GOTO 90
0590 REM---------------------------------
0600 REMSUB SHIRABERU
0610 IF S=1 THEN 750
0620 GOSUB 1100
0630 IF F<>1 THEN 680
```

図 8-3　ひろいものを解くプログラム　②

```
0640 PRINT "ｺﾀｴ:"
0650 GOSUB 1160
0660 S=1
0670 GOTO 750
0680 FOR D1=1 TO D
0690    GOSUB 950
0700    IF F=0 THEN 740
0710    GOSUB 770
0720    GOSUB 600
0730    GOSUB 860
0740    NEXT D1
0750 RETURN
0760 REM--------------------------------
0770 REMSUB TSUGI NO ISHI O HIROU
0780 N=N+1
0790 D9(N)=D1
0800 X=X+X8(D1)
0810 Y=Y+Y8(D1)
0820 IF B(X,Y)<>0 THEN 800
0830 B(X,Y)=N
0840 RETURN
0850 REM--------------------------------
0860 REMSUB MODOSU
0870 B(X,Y)=0
0880 D1=D9(N)
0890 X=X-X8(D1)
0900 Y=Y-Y8(D1)
0910 IF B(X,Y)<>N-1 THEN 890
0920 N=N-1
0930 RETURN
0940 REM--------------------------------
0950 REMSUB HIROERU KA
0960 F=0
0970 IF ABS(D9(N)-D1)=2 THEN 1080
0980 X1=X
0990 Y1=Y
1000 X1=X1+X8(D1)
1010 Y1=Y1+Y8(D1)
1020 IF X1<=0 THEN 1080
1030 IF X1>X3 THEN 1080
1040 IF Y1<=0 THEN 1080
1050 IF Y1>Y3 THEN 1080
1060 IF B(X1,Y1)<>0 THEN 1000
1070 F=1
1080 RETURN
1090 REM--------------------------------
1100 REMSUB OWARIKA
1110 F=0
1120 IF N<X3*Y3-N3 THEN 1140
1130 F=1
1140 RETURN
1150 REM--------------------------------
1160 REMSUB PRINT
1170 FOR X2=1 TO X3
```

図 8 - 3　ひろいものを解くプログラム　③

```
1180    FOR Y2=1 TO Y3
1190       IF B(X2,Y2)=-9 THEN 1220
1200       PRINT B(X2,Y2);
1210       GOTO 1230
1220       PRINT "      ";
1230       NEXT Y2
1240    PRINT
1250    NEXT X2
1260 PRINT "-------------------------------"
1270 RETURN
1280 END
```

図 8-3 ひろいものを解くプログラム ④

```
==ｺﾞｲｼ ﾊﾟｽﾞﾙ:  ﾋﾛｲﾓﾉ==
ﾓﾝﾀﾞｲ:
    0
    0
          0
-------------------------------
ｺﾚ ﾊ ﾃﾞｷﾅｲ
===============================
ﾓﾝﾀﾞｲ:
                0
          0     0     0
    0     0     0     0     0
          0     0     0
    0                       0
-------------------------------
ｺﾀｴ:
                1
         14     2    13
    8     9     3     4     5
         10    11    12
    7                       6
-------------------------------
===============================
ﾓﾝﾀﾞｲ:
    0
    0     0
    0     0     0
    0     0     0     0
-------------------------------
ｺﾀｴ:
    1
    2     3
    8     4     7
    9     5     6    10
-------------------------------
```

```
================================
ﾓﾝﾀﾞｲ:
                 0
           0           0
     0                       0
     0                       0
     0     0     0     0     0
     0                       0
--------------------------------
ｺﾀｴ:
                 1
          13          14
     7                       8
     6                       5
    11    12     2     3     4
    10                       9
--------------------------------
================================
ﾓﾝﾀﾞｲ:
```

図 8-4　ひろいものプログラムの動作結果

とはいうものの，問題として違う点も多いので，ひろいものと桂馬道の2つのプログラムを普通に作ると，それだけ手間はかかります。しかし，プログラムを構造的に設計してから作れば，2つ目のプログラムは，最初のプログラムの一部を使って非常にやさしくできることになります。

桂馬道のプログラムを作ったときには，うまくサブルーチンを作ることなどで，その手だてをしてきました。したがって，このひろいもののプログラムは，手間をかけずにできたわけです。

次にこのひろいものプログラムを，桂馬道プログラムと比べながら見てみましょう。

●問題の与え方

DATA文で，配列Bに値をREADすることによって行ないます。図8-3のように，

　　石があるところには0，
　　石がないところには－9

を入れておきます。

桂馬道プログラムの課題7―1（いろいろな図形をまわる）の考え

方と同じです。

●次にひろう石の方向

桂馬道プログラムの駒の進む方向と同じように配列Ｘ８，Ｙ８に，次にひろう石の方向を入れておきます。方向は**図8－5**のように四方向です。飛車のように，離れて動けるということは，プログラムで表わします。

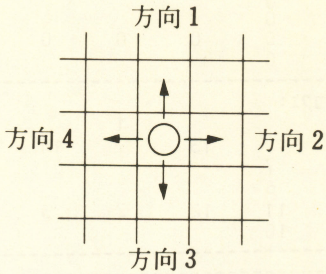


図8－5　次にひろう石の方向

●次の石をひろうサブルーチン

Ｄ１の方向へ進んで，石を１個ひろうサブルーチンを考えてみます。桂馬道のナイトが進むサブルーチンとだいたい同じですが，「現在の位置とひろう石の間が離れていても，間になにもなければよい」というところが違います。

盤の配列Ｂには，次のものがはいることなります。

- 最初から石がないところ……………………………－９
- まだ石があるところ…………………………………０
- もう石がひろわれてしまったところ
  (Ｎは何番目にひろったかを表わす)……………Ｎ（＞０）

そこで次の石の位置を計算するところで，次のようにIF文をつけ加えてループにします。

```
        ⋮
X＝X＋X８（D１）  ←──┐
Y＝Y＋Y８（D１）      │
IF  B（X, Y）<>0  THEN ─┘
        ⋮
```

こうすることは，Ｂ（Ｘ，Ｙ）が０でないかぎり，つまり石がないかぎり，位置を進む方向に動かすということになります。

●ある方向で石がひろえるかを調べるサブルーチン

このサブルーチンも，桂馬道プログラムの行けるかどうかを調べるサブルーチンとだいたい同じですが，少し変わったところがあります。

第1には，次の石をひろうサブルーチンと同じように，探す石の位置を求めていくところで，ループを作って，先の方まで見ていきます。

第2に，今来たばかりの方向へはもどれない，というルールがあります。このルールは，F＝0の行の後に

IF　ABS(D 9 (N)－D 1)＝2　THEN→(RETURNへ)

という1行を加えるだけで実現しています。なぜ，このルールがこれだけで表わせるのか考えてみましょう。D 9(N) は直前に進んだ方向です。D 1はこれから進む方向です。上の1行によってこのD 9 (N)とD 1の差が2のときにFを0（つまりD 1方向ではひろえないという意味）としてリターンすることになります。図8-5を見てみると，方向が1と3，2と4の組合せのときに，つまり方向の番号が2違いのときに互いに逆方向なります。したがって，D 9 (N) とD 1が逆方向，すなわち，Uターンするときには F＝0としてリターンしていきます。

●探索のサブルーチン

「調べる」というサブルーチンが，再帰的に縦型探索を行ないます。今度は1つ解が出てきたら，もうそれ以上は調べないような形にしました。

このことを行なうために変数Sを使い，Sが0なら答がまだ出ていないことを，1なら答が出たことを表わすようにします。このSのようなものを普通フラグといいます。ここで行なっていることは，

(1)　問題を解き始める前にSを0にする

(2)　解が出たらSを1にする

(3)　「調べる」サブルーチンは，Sが1ならばなにもしないですぐリターンする

(4)　メインではSが1なら次の問題に進む

ことです。こうすることで，解が多く出てこまるときに，1つしか出さないようにできるわけです。

●その他のサブルーチン

ひろった石を1つ元にもどすサブルーチンも，桂馬道のときと同じく必要です。これにも，

```
X＝X－X8（D1）
Y＝Y－Y8（D1）
IF　B（X, Y）<>N－1　THEN
```

のようにループを作り，直前の石の位置を調べます。

このほかに終了を判定するサブルーチンと，盤を表示するサブルーチンを作っておきます。これらは桂馬道の場合とほとんど同じです。

●プログラムのモジュール化は大事である

覆面算の章や桂馬道プログラムの章では，プログラムのする仕事をうまくサブルーチンに分けることが重要である，といいました。桂馬道のプログラムを作るとき，うまくサブルーチンに分けたので，今度ひろいもののプログラムを作るのがとてもやさしくなったのです。ここではうまくサブルーチンに分けることを，もう少し深く考えてみます。

プログラムの設計の立場からは，いろいろな機械のひとかたまりになった一部分のことをモジュールというように，サブルーチンのようなひとかたまりのものもモジュールといいます。プログラムを設計するときも機械を設計するときも，あるいは運動会のような大きな行事を行なうときも同じです。全体をうまくモジュールに分けることが大事なのです。

では，なぜモジュールに分けるのが大事なのでしょうか。それは普通の人間は非常に大きなもの（ものとはプログラム，機械，運動会，……なんでもそうですが）の全体を見とおすことができないからです。普通のBASICのプログラムならば，全体をすべて見とおせる大きさは多くて200行位のものでしょう。これより大きいプログラムでは，ある部分を見ていると他の部分のことを忘れてしまいがちになります。

見とおせる範囲の広い人たちがいないことはありません。この種類は2種あって，頭の構造が特殊な人々と，中学生や高校生ぐらいまでの

若い人々です。それにしても，数千行を一度に見とおすことは無理だと思います。そこでプログラムをモジュールに分けるわけです。

ある1つのモジュールを作ったり，見たりしているとき，ほかのモジュールの中で行なっている複雑な計算は忘れてしまってかまわないのです。ほかのモジュールについて知っていなければならないものは，その機能です。具体的には，

① そのモジュールは入力データでどんなものを必要とするか，

② 処理した結果の出力データがどこに出てくるか，

③ 他に影響を及ぼすとしたらどんなことか，

ということだけです。

つまり，作ったり見たりしているモジュール以外のものは**ブラックボックス**として見ればよいわけです。こうすることによって，見わたせる範囲が数十行位でも（私もこの程度です），数千行のプログラムを自由に書くことができます。

若い人たちの中には，モジュール化などしなくてもどんどんプログラムが書けると思う人がいるかもしれません。しかし次のことを考えてください。

(1) 年をとると見わたせる行数がへる。今から訓練すべきです。

(2) 社会に出ると，プログラムはたいてい共同して作る。わかりにくいプログラムは許されません。

(3) 自分のプログラムでも半年たてば忘れてしまう。日がたってからみつかったバグを取ろうとしたとき，非常に苦労します。

●モジュールの作り方

では，どのようにモジュールを作ればよいのでしょうか。ひとことでいえば，ひとかたまりのものをモジュールにすればよいのです。具体的にモジュール（BASICでは普通，サブルーチンにするのがよい）を作る指針をいくつかあげてみます。

(1) ひとかたまりの仕事をする部分を，1つのモジュールに分けます。ある仕事がひとかたまりかどうかはどう調べるかというと，「仕事が1つの文で書き表わせる」ということで判定するのがよ

いといわれています。たとえば,

(a)　局面を表示する

(b)　手を入力して変わった局面を表示する

のうち,(a)の仕事をするものは,モジュールにするのがよいでしょう。しかし(b)は手を入力する,というのと局面を表示するという2つの文でできているので,仕事はひとかたまりでないということになります。ですから(b)は,さらに2つに仕事を分けた方がいいことになります。

(2)　分け方の手順は,まず,全体をメインとなる1つのモジュールと数個のモジュールに分けます。さらにそのモジュールのうち,たくさんの手順を必要とするものは,さらに分割して下位のモジュールを作っていきます。

(3)　モジュールとモジュールの間では,なるべく引数によってデータを受け渡すのがよいのです。BASICのサブルーチンには,残念ながら引数はありません。そこで,BASICでは,あらかじめ決めておいた2,3の変数だけでサブルーチンとメイン（あるいは上位のサブルーチン）とデータを受け渡すのがよいでしょう。前に述べたように,こうすればモジュール（サブルーチン）の機能さえ知っていれば,その中のめんどうな処理を忘れてしまってもよいということになります。モジュールとモジュールの間では,なるべく結びつきを小さくするということです。

(4)　モジュールの中では,いろいろな結びつきは強い方がよいのです。結びつきが弱いということは,さらにモジュールに分けられるかもしれない,ということです。

(5)　1つのモジュールの大きさは,BASICのサブルーチンでは20行程度,つまりディスプレーの一画面にはいる大きさがよいと感じています（人によって感じ方は違うと思いますが)。

次にプログラムを作るときには,ぜひこの方法を使ってみることをおすすめします。大きいプログラム作成になるほど,その能率は下がってくるといわれていますが,モジュール化によってそれがかなり防げるでしょう。

課　題

8―1　次の石をひろいに行くとき，逆もどりできないだけでなく，必ず右か左に曲がらなければならない，というルールにするにはどうすればよいでしょうか。(解答201ページ)

8―2　次の石をひろいに行くとき，上下左右だけでなくななめにも動ける，というルールにするにはどうすればよいでしょうか。

つまりチェスのクィーンのように，動いて石をひろうということです。(解答201ページ)

8―3　次に石をひろいに行くとき，今まで石のあったところは通ってはいけない，というルールにするにはどうすればよいでしょうか。(解答201ページ)

8―4　次に石をひろいに行くとき，最初に石のあったところだけしか通れない，というルールにするにはどうすればよいでしょうか。(解答202ページ)

休憩室 8

100行のプログラム10個と1000行のプログラムを1個ではどちらが作りやすい？

普通のプログラムなら100行のプログラムを10個の方が作りやすいのです。だれでもそう感じています。ではなぜでしょうか？

それはプログラムの部分部分を設計して作っていくときに，**ほかの部分とのかかわりあい**を考えながら進めているからです。

100行のプログラムの一部分を作るときは，100行の大きさのものとのかかわりあいを考えればよいだけです。しかし，

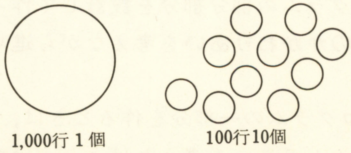
1000行のプログラムなら1000行の大きさのものとのかかわりあいを見ながら仕事をすることになります。したがって，かかわりあいについて考える量というのは，プログラムの大きさの二乗に比例します。100行のプログラムで，かかわりあいを100×100考えるとすれば，1000行では1000×1000考えることになります。結局，100行のプログラム10個に対する1000行のプログラム1個の間でのかかわりあいについて考える量の比は，

$$\frac{1000\times1000\times1}{100\times100\times10}=10$$

となり，1000行のプログラムの方が10倍大変であるということになります。作りやすさそのものについては，行数に比例する要因もあるでしょうから，これほど極端にはなりません。しかし，1000行プログラム1個の方がずっと大変であることは確かです。

ところが，1000行のプログラムがきれいに100行ずつ10個の部分に分かれてしまったらどうでしょうか？　そうすれば100行のプログラム10個と同じことになりますね。なるべくこのようにしようとするのが**モジュール化**なのです。モジュール（サブルーチン）の外についてはできるだけかかわりが少なくなるようにします。そうしてモジュールの中を作るときは，モジュールの中とだけのかかわりあいを考えればすむようにするわけです。

結局のところ，100行10個に対する1000行1個の作りにくさというのは，モジュール化することも考えに入れて，1.5～2倍に感じられるようです。



9★迷路パズル

囲碁の表示システムを作る

【問　題】

迷路の中の出発点から外へ出られるか調べてください。

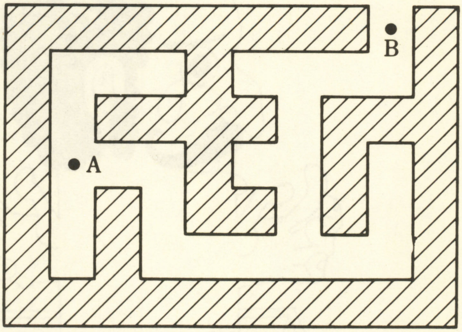


図 9－1　A点から出発して外へ出られるか調べる

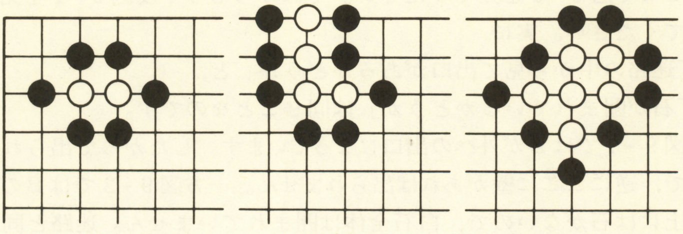
まずこの問題から始めてみましょう。人間ならば，こんな問題は5秒でできますが，計算機にやらせるとなるとちょっとしたテクニックが必要です。

また，計算機で迷路を解くということと囲碁とはかなり関係があります。上の問題を解くプログラムを作ってから，碁盤を表示する（つまり手を入力すると碁盤の代わりになって動く）プログラムを考えてみましょう。

●石の取りと迷路

囲碁で最も基本になっているルールは，「石の取り」です。**図 9－2**の(**a**)の白石は黒石に四方を囲まれているので，このようになったときにすぐ取り上げられます。しかし，図 9－2 の(**b**)の白石は取られません。おのおのすきまがあいているからです。

では，この(a)と(b)を計算機で分類するにはどうすればよいでしょうか。石がずっと遠くまでつながっていることもあります。



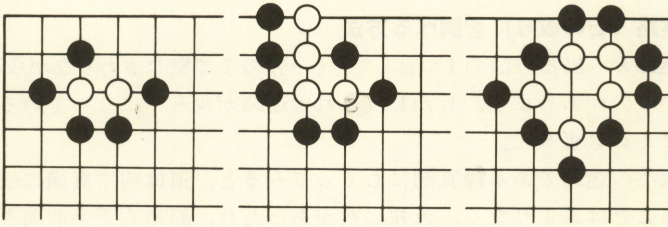


図 9-2　取られる石(a)と取られない石(b)

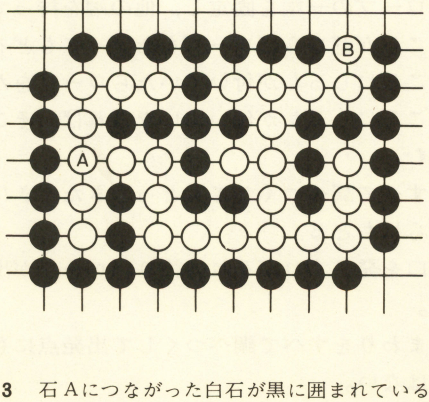


図 9-3　石 A につながった白石が黒に囲まれているか調べる

ここで**図９－３**を見てください。それからもう一度図９－１を見て比べてください。実に，

「迷路の中から見て出口があるかどうか」と，

「石が囲まれているかどうか」は同じことなのです。

図９－１ではＢが外への出口になっています。したがって出られるわけで，逆にここに壁があれば出られません。一方図９－３ではＢの白石の上には石がないので，白石全体は囲まれていません。迷路と同じように，Ｂが黒石だったら，白石は囲まれていることになります。

●迷路（石の取り）を調べる方法

迷路から出る方法の１つとして，「一方の手で壁にさわりながら進む」というのがあります。しかし，これは迷路がループになっていると，うまくいきません。

天然の迷路である鐘乳洞にもぐってみると，道は前や真横に続いていることはあまりなく，天井に穴があったり，斜め右下手前向きに割れ目があって先に続いていたりします。このように立体的な場合もこの方法ではうまくいきません。ちょっとした迷路には別の方法が必要になります。

そこで「ロープとしるし付け」による方法でやってみることにします。この概略を説明してみましょう。ルールは次のようになります。

(1)　出発点にロープの一端を固定し，他の端を持って始める。

(2)　進んだところが壁ならばロープをたどってもどる。

(3)　進んだところにしるしが付いていたらロープをたどってもどる。

(4)　進んだところにしるしが付いていなければ，そこにしるしを付け，先に進む。

(5)　まわりをすべて調べつくしてもどってきたときには，ロープをたぐってさらにもどる。

(6)　途中で出口を発見すればそれで終り。ロープが出る道すじの１つを表わす。

(7)　出発点のまわりをすべて調べつくして出発点にもどってきたときには出口はない。

この考え方を使って，迷路に出口があるかどうかを調べるアルゴリズムをフローチャートで書いてみると，**図 9-4** のようになります。アルゴリズムは途中で自分を使っています。つまり再帰的なわけです。

桂馬道やひろいものと同じように，これも縦型探索といえます。し

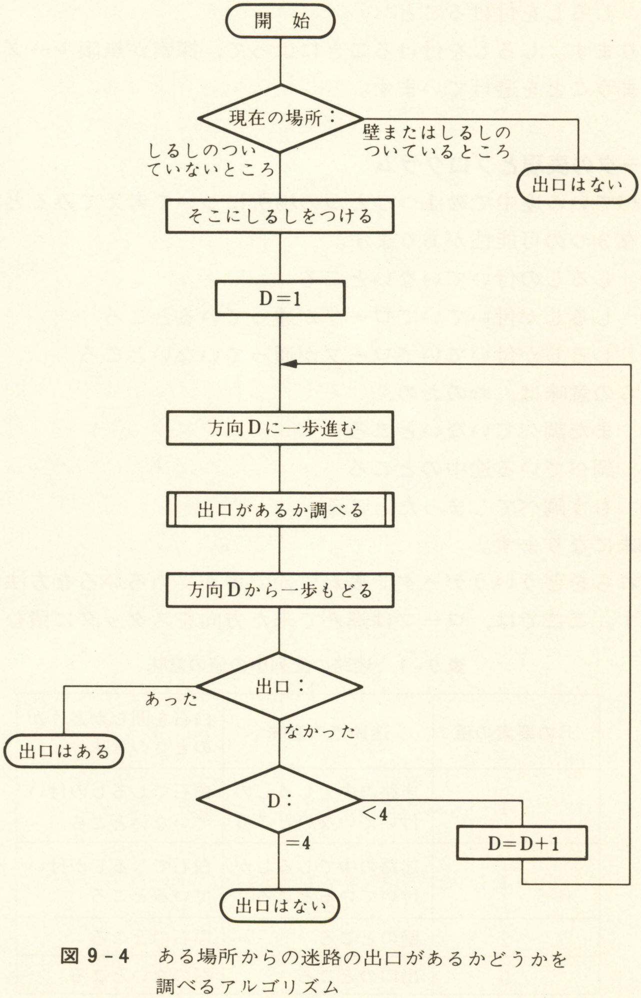


図 9-4　ある場所からの迷路の出口があるかどうかを調べるアルゴリズム

たがって，いろいろの点でよく似ています。桂馬道の場合と違う点の方を述べると，

① 動き方

② 終了の方式（全部まわったか，あるいは出口に出たか）

③ しるしを付けること

があります。しるしを付けることによって，探索が無限ループに落ちてしまうことを避けています。

●データの表現とプログラム

調べている途中での1つひとつの場所について考えてみると，次のような3つの可能性があります。

(a) しるしの付いていないところ

(b) しるしが付いていてロープが通っているところ

(c) しるしが付いていてロープが通っていないところ

これらの意味は，おのおの，

(a) まだ調べていないところ

(b) 調べている途中のところ

(c) もう調べてしまったところ

の意味になります。

これらをどういうデータで表わすかですが，いろいろな方法がありえます。ここでは，ロープは進んで来た方向をスタックに積むことに

表9-1 迷路の配列中の値の意味

Bの要素の値	迷路での意味	白石を囲むかどうかのときの意味
1	迷路の中でしるしの付いていないところ	白石でしるしの付いていないところ
－1	迷路の中でしるしが付いているところ	白石でしるしが付いているところ
2	壁のところ	黒石のところ
0	出口のところ	石のないところ

よって表わすことにします。それ以外については**表 9 - 1**のような数値を，迷路の図に対応する二次元配列の要素に格納します。また，石が囲まれているかどうかの問題と考えた場合の意味も書いておきます。

このような方式によって作ったプログラムを**図 9 - 5**に示します。出口がある（石が囲まれていない）場合，変数Tに0，そうでない場合には1を与えるようになっています。

```
0010 PRINT "ﾒｲﾛ ｶﾗ ﾃﾞﾗﾚﾙ ｶ"
0020 DIM B(19,19),X9(4),Y9(4)
0030 DIM S(1000),C$(3)
0040 READ C$(1),C$(2),C$(3)
0050 DATA " .", " *"," 0"
0060 READ X3,Y3
0070 FOR X=1 TO X3
0080    FOR Y=1 TO Y3
0090        READ B(X,Y)
0100        NEXT Y
0110    NEXT X
0120 DATA 8,10
0130 DATA 0,0,0,0,0,0,0,0,0,0
0140 DATA 2,2,2,2,2,2,2,2,1,2
0150 DATA 2,1,1,1,2,1,1,1,1,2
0160 DATA 2,1,2,2,2,2,1,2,2,2
0170 DATA 2,1,1,1,2,1,1,2,1,2
0180 DATA 2,1,2,1,2,2,1,2,1,2
0190 DATA 2,1,2,1,1,1,1,1,1,2
0200 DATA 2,2,2,2,2,2,2,2,2,2
0210 REM ISHI NO ICHI
0220 REM 4-HOUKOU
0230 FOR D=1 TO 4
0240    READ X9(D),Y9(D)
0250    NEXT D
0260 DATA 0,1,0,-1,1,0,-1,0
0270 P=0
0280 GOSUB 730
0290 I=1
0300 X=5
0310 Y=2
0320 GOSUB 400
0330 IF T=0 THEN 360
0340 PRINT "DERARENAI."
0350 STOP
0360 PRINT "DERARERU."
0370 GOSUB 730
0380 STOP
0390 REM------------------------------
0400 REM SUBR TORERUKA?
0410 IF X<1 GOTO 700
0420 IF Y<1 GOTO 700
0430 IF X>19 GOTO 700
0440 IF Y>19 GOTO 700
```

```
0450 IF B(X,Y)<0 GOTO 700
0460 IF B(X,Y)+I=3 GOTO 700
0470 IF B(X,Y)=I GOTO 560
0480 T=0
0490 REM MICHI O KAKU
0500 FOR P2=P TO 1 STEP -1
0510    X=X-X9(S(P2))
0520    Y=Y-Y9(S(P2))
0530    B(X,Y)=0
0540    NEXT P2
0550 RETURN
0560 B(X,Y)=-B(X,Y)
0570 FOR D=1 TO 4
0580    X=X+X9(D)
0590    Y=Y+Y9(D)
0600    P=P+1
0610    S(P)=D
0620    GOSUB 400
0630    D=S(P)
0640    P=P-1
0650    Y=Y-Y9(D)
0660    X=X-X9(D)
0670    IF T=1 GOTO 690
0680    RETURN
0690    NEXT D
0700 T=1
0710 RETURN
0720 REM-----------------------------------
0730 REM SUBR PRINT
0740 PRINT
0750 FOR X2=1 TO X3+1
0760    FOR Y2=1 TO Y3+1
0770       PRINT C$(ABS(B(X2,Y2))+1);
0780       NEXT Y2
0790    PRINT
0800    NEXT X2
0810 RETURN
0820 END
```

図 9-5 迷路から出られるかどうか（石が囲まれているかどうか）を判定するプログラム

●迷路から出る道

この問題を考えるとき，迷路から出られるかどうかがわかるだけでは，あまり面白くありません。碁石の取りの判定と考えるならば，このままでいいですが，迷路だと考えるならば，どうしても「迷路から出る道」も出力できるようにしたいところです。

出口に来たときには，ロープが通っているところが迷路から出る道

の1つになっています。ロープの情報はスタックの中にあります。スタックには進んで来た方向がはいっているので，これを出力すれば道はわかります。しかし，次のようにすればもっとわかりやすいでしょう。つまり，スタックの中身に従って配列に書き込んでいきます。

なお，道の出力は，出口が見つかったときにすぐ行なう必要があります。サブルーチンからリターンした後では，スタックがポップ・アップしてしまっています。

これを実現するにはサブルーチンの中の，

```
    T＝0
```

の行の後に，

```
  FOR P2＝P TO 1 STEP －1
    X＝X－X9(S(P2))
    Y＝Y－Y9(S(P2))
    B(X,Y)＝0
    NEXT P2
  GOSUB ――→ (プリント サブルーチンへ)
```

とすればよいでしょう。こうすると結果は，**図 9-6** のようになります。

```
メイロ カラ デ゛ラレル カ

 . . . . . . . . . . .
 0 0 0 0 0 0 0 0 * 0 .
 0 * * * 0 * * * * 0 .
 0 * 0 0 0 0 * 0 0 0 .
 0 * * * 0 * * 0 * 0 .
 0 * 0 * 0 0 * 0 * 0 .
 0 * 0 * * * * * * 0 .
 0 0 0 0 0 0 0 0 0 0 .
 . . . . . . . . . . .
DERARERU.

 . . . . . . . . . . .
 0 0 0 0 0 0 0 0 . 0 .
 0 * * * 0 * . . . 0 .
 0 * 0 0 0 0 . 0 0 0 .
 0 . . . 0 * . 0 * 0 .
 0 * 0 . 0 0 . 0 * 0 .
 0 * 0 . . . . * * 0 .
 0 0 0 0 0 0 0 0 0 0 .
 . . . . . . . . . . .
```

図 9-6　迷路から出る道の出力結果

●碁盤の表示システム

さて，迷路から出られるかどうか（石が囲まれているかどうか）を判定することができることになったので，次に碁盤の表示システムを考えてみましょう。詳しく述べると，

> 碁盤表示システム
> - 画面に碁盤を表示する。
> - 手を入力すると，禁手かどうか判定し，禁手ならやりなおさせる。
> - 禁手でなければそれを盤上につけ加え，取られる石があ

れば取り除く。

●「待った」の入力があれば手をもどす。

というようなもので，碁盤の代わりになり，石の取りの管理をします。

●囲まれている石を取り除く

囲まれているかどうかはわかったので，今度は囲まれている石を取り除くことを考えます。このことは「つながっている石を取り除く」ことで行ないます。

このアルゴリズムを**図9-7**に示します。これは引数を2個（R1とR2）持っていて，**R1のつながりをR2のつながり**に変えます。白石を取り除くときにはR1を1，R2を0とします。

また，このアルゴリズムを，囲まれているかどうかの判定のときに付けたしるしを消すのにも使います。このときはR1を－1，R2を1とします。

さらに，このアルゴリズムは「待った」のときに石をもどすことに使えます。

このように，一見違う仕事でも，よく考えるとほとんど同じことであることがよくあります。ここではR1のつながりをR2のつながりに変える，ということで全部できるようにしています。つまり，よいプログラムを書くには，次のことが大事なのです。

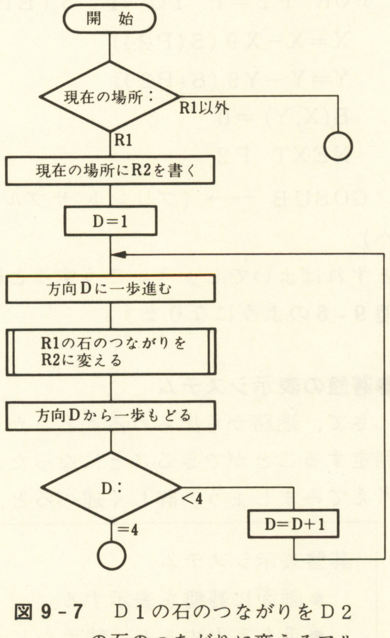


図9-7　D1の石のつながりをD2の石のつながりに変えるアルゴリズム

似た部分を観察し
共通の機能を発見して
ひとつのモジュールにまとめよう

●禁手かどうかの判定

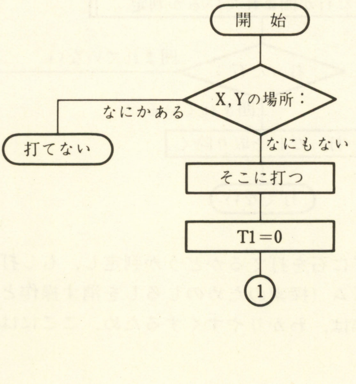
ここまでで囲まれた石を取り除けるようになったので，ここで禁手の処理を考えます。囲碁のルールの中心となる部分は次のようにまとめられるでしょう。

(1)　石を打つと，その石で囲みができ上がる相手の石は取り除かれる。

(2)　石を打つことにより，その石につながった石全体が，囲まれることになるとき，その手は禁手である。

(3)　ただし，(2)において(1)によって相手の石が取り除かれることがあるときは除外される。

これをアルゴリズムとして実現したのが，**図 9－8** です。この図では手をもどすためにスタックに積む操作などの細かい点は除いて書いてあります。また，ルールの１つとして重要な「劫(コウ)」についてもここでは書いてありませんが，これは課題として出すことにします。



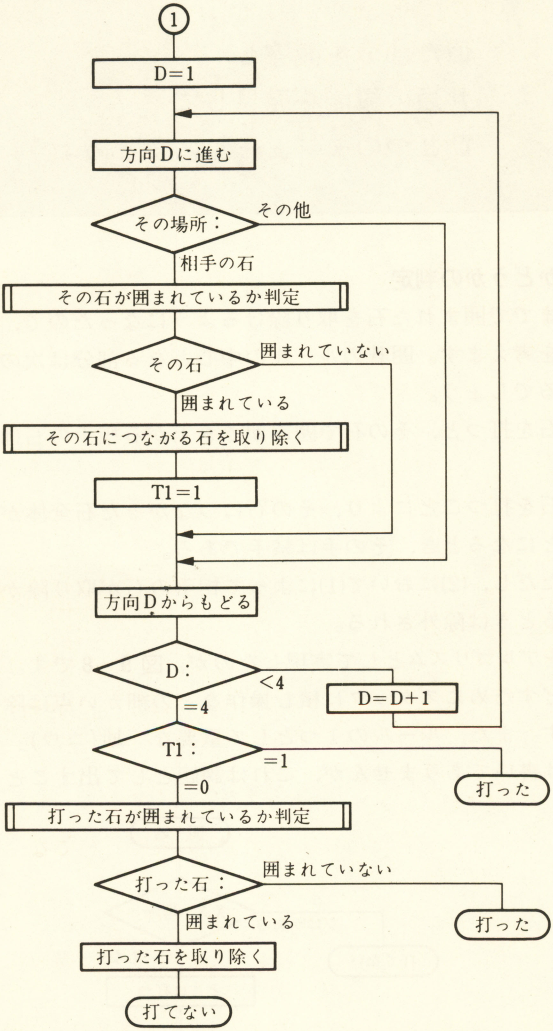


図 9-8　X，Yに石を打てるかどうか判定し，もし打てるならば打つアルゴリズム（探索のためのしるしを消す操作と，手をスタックに積む操作は，わかりやすくするため，ここには書いてありません）

●「待った」をありにする

待ったをできるようにしておくわけは２つあります。

第１には次の章で詰碁を解くためのプログラムの道具を準備する意味があるからです。待ったは手をもどすことです。つまり探索作業の中の後もどり（backtracking）の操作にあたるわけで，ここでそれを作っておけば，後で役に立つのです。

第２には，使い勝手の問題があります。待ったがないと，ちょっとでも石の打つ位置を間違えたときに困ってしまいます。待ったというのはよくないし，上達のためにはならないからいらない，と考える人もあるかもしれません。しかし，実際に碁の対局を見ていると，石を打ってからちょっと動かすなどというのがときどきあることがわかります。

このように人間が使うシステムを作るときには，現実を分析し，人間にとっての使いやすさをよく考えておく必要があります。

人間とのインターフェースを
よく考えて！

●後もどりの実現

後もどりを実現するためには打った手の位置だけでなく，取った石の情報も保存しておく必要があります。

プログラムでは，これらの情報をスタックＳ１に積み上げています。後もどり操作とは，これらの情報を使って１手前の局面にもどすことです。

図 9-9 の手を打つときには，**図 9-10** のような情報がスタックに積まれます。取られる石については，全部の石ではなく根元の石だけをスタックに積みます。残りは図９-７のアルゴリズムでたどればよいのです。

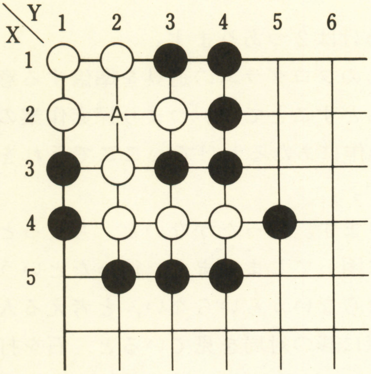


図 9－9　黒の手(A)

図では取られる石は３つの部分に分れます。根元の石も３つになります。

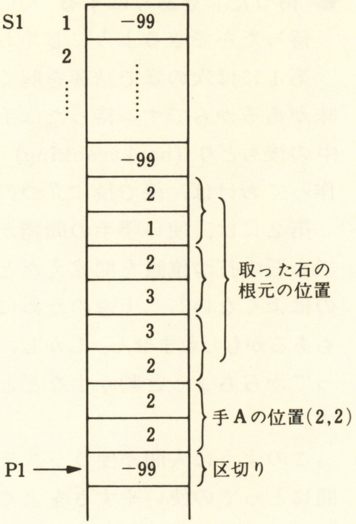


図 9－10　手Aによってスタック S 1 に積まれる情報

●碁盤表示システム

最後にプログラムを**図 9－11**に示します。また，ゲームの様子は，**図 9－12**のように出てきます。

```
0010 PRINT "<イゴ ヒョウジ システム>"
0020 REM MATTA ARI
0030 DIM B(19,19),X9(4),Y9(4)
0040 DIM  S(500),S1(1500),C$(3)
0050 REM TORISTK YOMISTK
0060 REM S1 STACK NO TE NO DATA
0070 REM X1,Y1,...,XN,YN,X,Y,-99
0080 REM TORU ICHI,UTSU ICHI,MARK
0090 P=0
0100 P1=0
0110 P1=P1+1
0120 S1(P1)=-99
0130 READ C$(1),C$(2),C$(3)
0140 DATA " •"," *"," 0"
```

図 9－11　碁盤表示システムのプログラム　①

```
0150 FOR X=1 TO 19
0160    FOR Y=1 TO 19
0170       B(X,Y)=0
0180       NEXT Y
0190    NEXT X
0200 REM ISHI NO ICHI
0210 REM 4-HOUKOU
0220 FOR D=1 TO 4
0230    READ X9(D),Y9(D)
0240    NEXT D
0250 DATA 0,1,0,-1,1,0,-1,0
0260 I=1
0270 GOSUB 1610
0280 PRINT C$(I+1);" ﾉ ﾊﾞﾝ, X,Y";
0290 INPUT X,Y
0300 IF X<0 THEN 370
0310 GOSUB 420
0320 IF U=1 THEN 350
0330 PRINT"ｿｺ ﾆﾊ ｳﾃﾅｲ"
0340 GOTO 280
0350 I=3-I
0360 GOTO 270
0370 PRINT "ﾏｯﾀ"
0380 GOSUB 1230
0390 I=3-I
0400 GO TO 270
0410 REM----------------------------------
0420 REM SUBR X,Y NI I O UTERUKA?
0430 REM KANOU NARA UTSU
0440 U=0
0450 IF B(X,Y)<>0 GOTO 850
0460 B(X,Y)=I
0470 T1=0
0480 FOR D=1 TO 4
0490    X=X+X9(D)
0500    Y=Y+Y9(D)
0510    P=P+1
0520    S(P)=D
0530    IF X<1 THEN 720
0540    IF Y<1 THEN 720
0550    IF X>19 THEN 720
0560    IF Y>19 THEN 720
0570    IF B(X,Y)+I<>3 THEN 720
0580    I=3-I
0590    GOSUB 960
0600    R1=-I
0610    R2=I
0620    IF T=0 THEN 700
0630    REM TOKERU
0640    P1=P1+1
0650    S1(P1)=X
0660    P1=P1+1
0670    S1(P1)=Y
0680    R2=0
```

図 9-11 碁盤表示システムのプログラム ②

```
0690     T1=T1+1
0700     GOSUB 1410
0710     I=3-I
0720     D=S(P)
0730     P=P-1
0740     Y=Y-Y9(D)
0750     X=X-X9(D)
0760     NEXT D
0770 IF T1>0 THEN 860
0780 GOSUB 960
0790 R1=-I
0800 R2=I
0810 GOSUB 1410
0820 U=1-T
0830 IF U=1 THEN 860
0840 B(X,Y)=0
0850 RETURN
0860 U=1
0870 P1=P1+1
0880 S1(P1)=X
0890 P1=P1+1
0900 S1(P1)=Y
0910 P1=P1+1
0920 S1(P1)=-99
0930 PRINT
0940 RETURN
0950 REM----------------------------------
0960 REM SUBR TORERUKA?
0970 IF X<1 GOTO 1200
0980 IF Y<1 GOTO 1200
0990 IF X>19 GOTO 1200
1000 IF Y>19 GOTO 1200
1010 IF B(X,Y)<0 GOTO 1200
1020 IF B(X,Y)+I=3 GOTO 1200
1030 IF B(X,Y)=I GOTO 1060
1040 T=0
1050 RETURN
1060 B(X,Y)=-B(X,Y)
1070 FOR D=1 TO 4
1080     X=X+X9(D)
1090     Y=Y+Y9(D)
1100     P=P+1
1110     S(P)=D
1120     GOSUB 960
1130     D=S(P)
1140     P=P-1
1150     Y=Y-Y9(D)
1160     X=X-X9(D)
1170     IF T=1 GOTO 1190
1180     RETURN
1190     NEXT D
1200 T=1
1210 RETURN
1220 REM----------------------------------
```

図 9-11 碁盤表示システムのプログラム ③

```
1230 REM SUBR TE O MODOSU
1240 P1=P1-1
1250 Y2=S1(P1)
1260 P1=P1-1
1270 X2=S1(P1)
1280 P1=P1-1
1290 IF S1(P1)=-99 THEN 1380
1300 R1=0
1310 R2=I
1320 Y=S1(P1)
1330 P1=P1-1
1340 X=S1(P1)
1350 P1=P1-1
1360 GOSUB 1410
1370 GOTO 1290
1380 B(X2,Y2)=0
1390 RETURN
1400 REM-----------------------------------
1410 REM SUBR ZUKEI-NURIKAE:R1>R2 AT(X,Y)
1420 IF X<1 THEN 1590
1430 IF Y<1 GOTO 1590
1440 IF X>19 GOTO 1590
1450 IF Y>19 GOTO 1590
1460 IF B(X,Y)<>R1 THEN 1590
1470 B(X,Y)=R2
1480 FOR D=1 TO 4
1490    X=X+X9(D)
1500    Y=Y+Y9(D)
1510    P=P+1
1520    S(P)=D
1530    GOSUB 1410
1540    D=S(P)
1550    P=P-1
1560    Y=Y-Y9(D)
1570    X=X-X9(D)
1580    NEXT D
1590 RETURN
1600 REM-----------------------------------
1610 REM SUBR PRINT
1620 PRINT "19181716151413121110";
1630 PRINT " 9 8 7 6 5 4 3 2 1 X / Y"
1640 FOR Y2=1 TO 19
1650    FOR X2=19 TO 1 STEP -1
1660       C1$=C$(B(X2,Y2)+1)
1670       IF C1$<>" •" THEN 1710
1680       IF (X2-4)<>INT((X2-4)/6)*6 THEN 1710
1690       IF (Y2-4)<>INT((Y2-4)/6)*6 THEN 1710
1700       C1$=" +"
1710       PRINT C1$;
1720       NEXT X2
1730    PRINT Y2
1740    NEXT Y2
1750 RETURN
1760 END
```

図 9-11　碁盤表示システムのプログラム　④

```
<イゴ" ヒョウジ" システム>
19181716151413121110 9 8 7 6 5 4 3 2 1 X / Y
. . . . . . . . . . . . . . . . . . .      1
. . . . . . . . . . . . . . . . . . .      2
. . . . . . . . . . . . . . . . . . .      3
. . . + . . . . . + . . . . . + . . .      4
. . . . . . . . . . . . . . . . . . .      5
. . . . . . . . . . . . . . . . . . .      6
. . . . . . . . . . . . . . . . . . .      7
. . . . . . . . . . . . . . . . . . .      8
. . . . . . . . . . . . . . . . . . .      9
. . . + . . . . . + . . . . . + . . .     10
. . . . . . . . . . . . . . . . . . .     11
. . . . . . . . . . . . . . . . . . .     12
. . . . . . . . . . . . . . . . . . .     13
. . . . . . . . . . . . . . . . . . .     14
. . . . . . . . . . . . . . . . . . .     15
. . . + . . . . . + . . . . . + . . .     16
. . . . . . . . . . . . . . . . . . .     17
. . . . . . . . . . . . . . . . . . .     18
. . . . . . . . . . . . . . . . . . .     19
* ﾉ ﾊﾞﾝ, X,Y?3,4
```

```
19181716151413121110 9 8 7 6 5 4 3 2 1 X / Y
. . . . . . . . . . . . . . . . . . .      1
. . . . . . . . . . . . . . . . . . .      2
. . . . . . . . . . . . . . . . . . .      3
. . . + . . . . . + . . . . . + * . .      4
. . . . . . . . . . . . . . . . . . .      5
. . . . . . . . . . . . . . . . . . .      6
. . . . . . . . . . . . . . . . . . .      7
. . . . . . . . . . . . . . . . . . .      8
. . . . . . . . . . . . . . . . . . .      9
. . . + . . . . . + . . . . . + . . .     10
. . . . . . . . . . . . . . . . . . .     11
. . . . . . . . . . . . . . . . . . .     12
. . . . . . . . . . . . . . . . . . .     13
. . . . . . . . . . . . . . . . . . .     14
. . . . . . . . . . . . . . . . . . .     15
. . . + . . . . . + . . . . . + . . .     16
. . . . . . . . . . . . . . . . . . .     17
. . . . . . . . . . . . . . . . . . .     18
. . . . . . . . . . . . . . . . . . .     19
0 ﾉ ﾊﾞﾝ, X,Y?16,16
```

```
19181716151413121110 9 8 7 6 5 4 3 2 1 X / Y
. . . . . . . . . . . . . . . . . . .      1
. . . . . . . . . . . . . . . . . . .      2
. . . . . . . . . . . . . . . . . . .      3
. . . + . . . . . + . . . . . + * . .      4
. . . . . . . . . . . . . . . . . . .      5
. . . . . . . . . . . . . . . . . . .      6
. . . . . . . . . . . . . . . . . . .      7
. . . . . . . . . . . . . . . . . . .      8
. . . . . . . . . . . . . . . . . . .      9
. . . + . . . . . + . . . . . + . . .     10
. . . . . . . . . . . . . . . . . . .     11
. . . . . . . . . . . . . . . . . . .     12
. . . . . . . . . . . . . . . . . . .     13
. . . . . . . . . . . . . . . . . . .     14
. . . . . . . . . . . . . . . . . . .     15
. . . 0 . . . . . + . . . . . + . . .     16
. . . . . . . . . . . . . . . . . . .     17
. . . . . . . . . . . . . . . . . . .     18
. . . . . . . . . . . . . . . . . . .     19
* ﾉ ﾊﾞﾝ, X,Y?16,4
```

```
19181716151413121110 9 8 7 6 5 4 3 2 1 X / Y
. . . . . . . . . . . . . . . . . . .      1
. . . . . . . . . . . . . . . . . . .      2
. . . . . . . . . . . . . . . . . . .      3
. . . * . . . . . + . . . . . + * . .      4
. . . . . . . . . . . . . . . . . . .      5
. . . . . . . . . . . . . . . . . . .      6
. . . . . . . . . . . . . . . . . . .      7
. . . . . . . . . . . . . . . . . . .      8
. . . . . . . . . . . . . . . . . . .      9
. . . + . . . . . + . . . . . + . . .     10
. . . . . . . . . . . . . . . . . . .     11
. . . . . . . . . . . . . . . . . . .     12
. . . . . . . . . . . . . . . . . . .     13
. . . . . . . . . . . . . . . . . . .     14
. . . . . . . . . . . . . . . . . . .     15
. . . 0 . . . . . + . . . . . + . . .     16
. . . . . . . . . . . . . . . . . . .     17
. . . . . . . . . . . . . . . . . . .     18
. . . . . . . . . . . . . . . . . . .     19
0 ﾉ ﾊﾞﾝ, X,Y?3,17
```

```
19181716151413121110 9 8 7 6 5 4 3 2 1 X / Y
. . . . . . . . . . . . . . . . . . .      1
. . . . . . . . . . . . . . . . . . .      2
. . . . . . . . . . . . . . . . . . .      3
. . . * . . . . . + . . . . . + * . .      4
. . . . . . . . . . . . . . . . . . .      5
. . . . . . . . . . . . . . . . . . .      6
. . . . . . . . . . . . . . . . . . .      7
. . . . . . . . . . . . . . . . . . .      8
. . . . . . . . . . . . . . . . . . .      9
. . . + . . . . . + . . . . . + . . .     10
. . . . . . . . . . . . . . . . . . .     11
. . . . . . . . . . . . . . . . . . .     12
. . . . . . . . . . . . . . . . . . .     13
. . . . . . . . . . . . . . . . . . .     14
. . . . . . . . . . . . . . . . . . .     15
. . . 0 . . . . . + . . . . . + . . .     16
. . . . . . . . . . . . . . . . 0 . .     17
. . . . . . . . . . . . . . . . . . .     18
. . . . . . . . . . . . . . . . . . .     19
* ﾉ ﾊﾞﾝ, X,Y?9,4
```

```
19181716151413121110 9 8 7 6 5 4 3 2 1 X / Y
. . . . . . . . . . . . . . . . . . .      1
. . . . . . . . . . . . . . . . . . .      2
. . . . . . . . . . . . . . . . . . .      3
. . . * . . . . . + * . . . . + * . .      4
. . . . . . . . . . . . . . . . . . .      5
. . . . . . . . . . . . . . . . . . .      6
. . . . . . . . . . . . . . . . . . .      7
. . . . . . . . . . . . . . . . . . .      8
. . . . . . . . . . . . . . . . . . .      9
. . . + . . . . . + . . . . . + . . .     10
. . . . . . . . . . . . . . . . . . .     11
. . . . . . . . . . . . . . . . . . .     12
. . . . . . . . . . . . . . . . . . .     13
. . . . . . . . . . . . . . . . . . .     14
. . . . . . . . . . . . . . . . . . .     15
. . . 0 . . . . . + . . . . . + . . .     16
. . . . . . . . . . . . . . . . 0 . .     17
. . . . . . . . . . . . . . . . . . .     18
. . . . . . . . . . . . . . . . . . .     19
0 ﾉ ﾊﾞﾝ, X,Y?
```

図 9-12　碁盤表示の様子

課　題

9—1　**図 9-13**はヘックスというゲームの１つの局面です。辺から辺へ石をつなげると勝になります。ＡからＢへつながっているかどうか調べるプログラムは，どうなりますか。(解答202ページ)

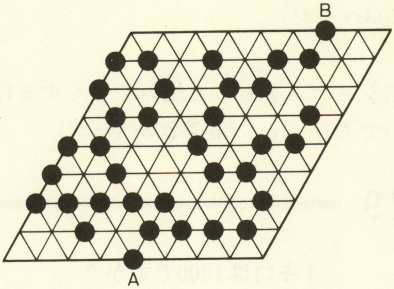


図 9-13　ヘックスの盤の上のつながり

9—2　右折しないで迷路から出る，というのはどのように解かせればよいでしょうか。たとえば**図 9-14**のような場合です。ヒント：迷路を三次元的に考えます。(解答202 ページ)

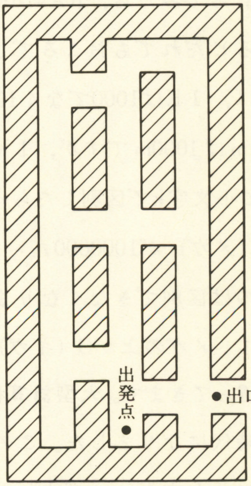


図 9-14　右折しないで出る迷路 ➡

9—3　碁盤表示システムで，あげはま（取った石の数）を表示するようにしてください。(解答203ページ)

9—4　碁盤表示システムで劫の禁手が打てないように改良してください。劫とは1つの石を取ったところへ1つの石を取りかえすことをいいます。これがないと無限の繰り返しになり，ゲームが成り立

ちません。(解答203ページ)

9—5　基盤表示システムをもっと使いやすくするには，どのように改良すればよいでしょうか。(解答203ページ)

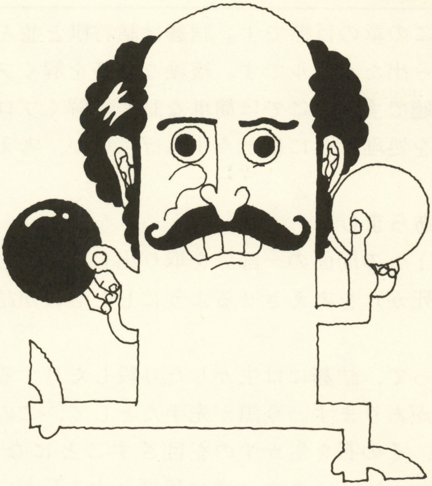
休憩室 9

1キロは1000ですか？

64KB（キロバイト）RAMを装備したというパソコンを買ってみたら，RAMが64000バイトしかなかった，となるとだれでもおこるでしょうね。コンピュータの世界では，1Kは1000でなく1024として使うのが普通です。1kmは1000mですが，1KBは1024Bです。これを大文字Kと小文字kで区別したらいいという人もいます。でも1M（メガ）が1000000だったり1024^2＝1048576だったりするのは区別できなくなります。mはミリになってしまいます。メガの上がG（ギガ）で，こうなると7％以上もくい違ってきます。大型計算機のディスクの容量はギガで計る時代になりました。この辺をちゃんと標準として決めておいた方がいいですね。

マイコンの雑誌の交換欄などに100K円などと書いてあるのは非常に気持悪く感じます。2400円はどうするのでしょうか。

10★詰碁を解く



詰碁を解く

これがこの章の目標です。詰碁は詰将棋と並んで，日本の伝統的なゲームから出たパズルです。複雑な詰碁を解くプログラムを作るのは大変な問題です。ここでは簡単な詰碁を解くプログラムを作り，知的なゲームを処理するにはどうすればよいか，考えたいと思います。

●詰碁のあらまし

囲碁の１つの局面の一部分を取り出し，石がとられるかとられないか（生か死か）を考えさせるようにしたものが詰碁のはじまりでしょう。

したがって，詰碁には生かしたり殺したりする目標になる石（のかたまり）があります。今黒が先手だとして，この目標の石も黒であるとすると，その石を生かすのを目ざすことになります。この場合を「黒先黒生」といいます。逆に目標とする石が白ならば，それを取ることを目ざします。これを「黒先白死」といいます。このほかに劫を目ざす場合もあります。

●勝っているか負けているかを表わす

黒先黒生ならば，目標の石が取られてしまったら，黒にとって負けている（失敗である）わけです。また白に打つ手がなくなってしまったら勝った（成功した）ことになります。これをまとめると**表10－1**のようになります。

さらに黒先黒生では，最終段階だけではなく，途中の段階でも，そ

表10－1　勝負の判定法

	黒の勝（白の負）	黒の負（白の勝）
黒先黒生	白は打つ手がない	目標の石が取られた
黒先白死	目標の石を取った	黒は打つ手がない

れから先どうやっても石が取られてしまうことになるなら，その段階も負けということができます。それから先，白がどう打っても石を取れないなら勝となります。黒先白死でも同じようなことがいえます。

このようにして決められる勝と負を，○×や変数Vの値で，

勝——○，V＝1

負——×，V＝−1

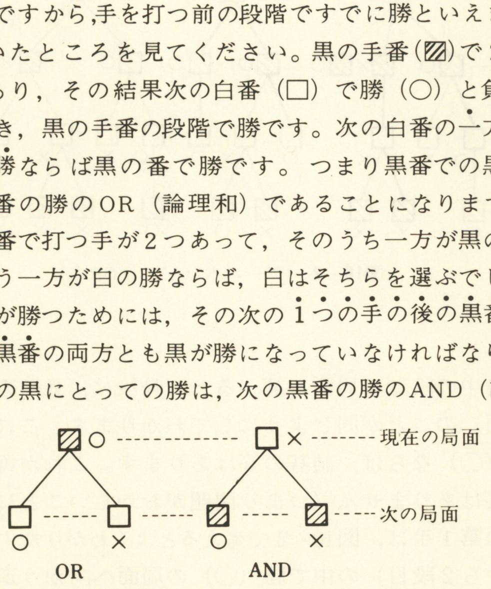
のように表わすことにします。黒から見た勝と負と，白から見た勝と負けはちょうど逆になることに注意してください。

●途中の段階の勝と負けの調べ方

最終段階の勝と負けは表10-1で決まりますが，途中の段階を決めなければなりません。今，黒の番で，打てる手が2つあるとします。この場合，2つの手のうち1つでも勝になる手があれば，その手を打てばいいのですから,手を打つ前の段階ですでに勝といえます。**図10-1**のORと書いたところを見てください。黒の手番(▨)で2つの黒の手（斜線）があり，その結果次の白番（□）で勝（○）と負（×）になっているとき，黒の手番の段階で勝です。次の白番の一方で勝**または**もう一方で勝ならば黒の番で勝です。つまり黒番での黒にとっての勝は次の白番の勝のOR（論理和）であることになります。

逆に，白番で打つ手が2つあって，そのうち一方が黒の勝になっていても，もう一方が白の勝ならば，白はそちらを選ぶでしょう。

今度は黒が勝つためには，その次の1つの手の後の黒番**と**もう1つの手の後の黒番の両方とも黒が勝になっていなければなりません，つまり白番での黒にとっての勝は，次の黒番の勝のAND（論理積）であ



（▨：黒番，□：白番；○：黒にとっての勝，×：黒にとっての負）

図10－1 ANDとOR

ることになります。

さらに何手も先まで調べるときには，このANDとORがたがいちがいに現れます。詰碁などの1つの局面を完全に分析してみると結局，**図10-2**のような木状の図になります。このような木をゲーム木（あるいはAND／OR木）といいます。おのおのの局面（▨や□）で下になにもつながっていないところは，そこで勝と負の判定を（それ以上進まずに)すぐできるところです。下になにかつながっている局面では，その勝負は下につながっている局面の勝負のANDかORで調べることができます。

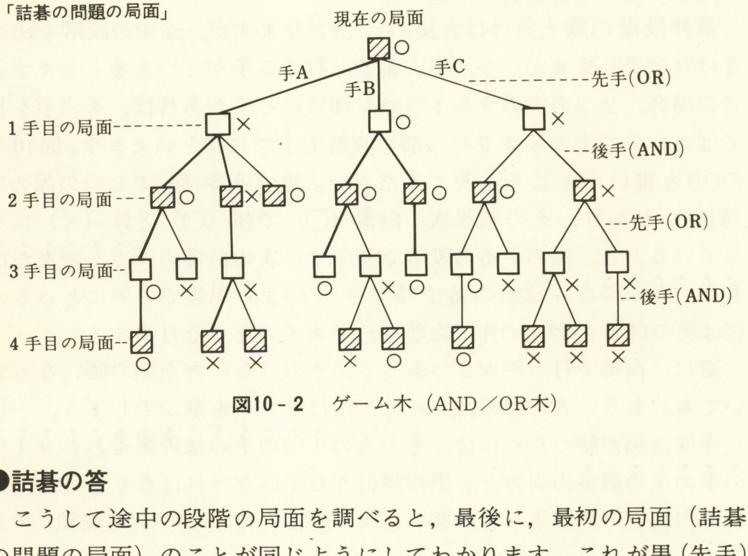


図10-2　ゲーム木（AND／OR木）

●詰碁の答

こうして途中の段階の局面を調べると，最後に，最初の局面（詰碁の問題の局面）のことが同じようにしてわかります。これが黒（先手）にとって勝（○）ならば，詰碁の答はあります。これが負（×）ならば，詰碁の答はありません（つまり問題がおかしいことになります)。

詰碁の答の第1手は，図10-2で考えるとよくわかりますが，図の第1手目（上から2段目）の中で勝（○）の局面へ向かう道です。図では手Bです。手Aや手Cではうまくいきません（×印でわかるわけです)。この第1手目をさがすことがもっとも大事なことです。

詰碁の答の第2手は，ルールとして厳密には決まっていません（詰将棋なら，手順が最長になる手を選びます)。実際には3手目より後にいい手が出るように選ばれます。しかしプログラムによって，客観的に判定できない「いい手」を選ぶことはできません。そこでプログラムでは簡単に，全部表示する，ということにします。

3手目より後は，偶数手目は2手目，奇数手目は1手目と同じことになります。

●探索と枝刈り

やはり今までの問題と同じく縦型探索を行ないます。図では手Aの後の局面を全部調べ，その後で手Bの局面を全部調べ，……というようにします。

ここで手Bと手Cのところを見てください。手Bを調べると結局，手Bを打つと勝つということがわかりました。そうすると手Cより後の局面は全然調べなくてよいのです。このようなところは，他にもいくつかあります。図では，左側にある手を先に調べ始めることにした場合に，調べることになる部分を太線で書いてあります。調べなくてすむ部分は細線です。

このようにすれば，全部の手を調べるのに比べて，ずっと効率よく（つまり計算時間を短く）することができます。この得は複雑な問題になればなるほど大きくなります。前に述べた組合せ論的爆発を先に延ばすことができます。このように全部の枝を調べなくてすむようにすることを「枝刈り」ということがあります。

勝(○)と負(×)を数として＋1と－1とすれば,ORは最大値(maximum)をとることに，ANDは最小値（minimun）をとることにあたります。このようにすると，次のような場合を含めてうまくいきます。つまり，探索の木が大き過ぎて全部は調べられないとき，途中の局面で勝負の程度を判定してしまいます。たとえば勝ちそうであれば0.7，少し負けそうであれば－0.2というようにします。このとき，AND／OR探索の代わりにミニマックス探索ということがあります。

交代して手を指す2人のゲームはたいていこの考えが，勝，負を調

べ，正しい手を見つけることの基本になっています。チェス，チェッカー，オセロではこれをおしすすめて，人間より強いプログラムが作られるようになっています。また，囲碁や将棋でも少しずつ人間と対戦できるプログラムの研究が進められています。

●問題の与え方

さて,いよいよ勝負判定をどうするかを考えてみます。今まで述べてきたことを，アルゴリズムとして実現すればよいのです。碁盤表示システムのところで必要な道具は全部そろえました。ここでは中心の部分だけを作れば完成します。

アルゴリズムの前に，問題の与え方を述べておきます。問題は，盤面の石の配置によって与えます。おのおのの盤面の位置に対して，

白石を　２，

黒石を　１，

空所を　０，

の値で与えます。また，同時に石を打つ範囲を与えます。石を打つ範囲を与えなければ，とんでもない遠い所へ石を打つことも調べなければならないことになり，ほとんど計算不可能になります。そこで問題の周辺の石の打てる領域を決めてやります，人間ならばこういうことはあたりまえでわかっていますが，プログラムが打って意味のある領域を計算するというのは，かなりむずかしい問題です。ここでは簡単にするためにデータで与えます。石を打つ範囲の位置は上の０，１，２の値に対しおのおの10をたし，10，11，12で表わすことにします。

調査する手は，黒，白とも，この範囲の中に打つ手全部と，手ぬき(パス）することです。

●勝負判定のアルゴリズム

あるひとつの局面を入力として，その次に打つ側から見た勝負を判定する（すなわち変数Ｖに＋１か－１を出力する）アルゴリズムを考えてみます。

まず，終了の条件はどんなものがあるかというと，

①　目標の石が取られてしまった

②　白，黒の両者が続けてパスした

の2つがありえます。最終的なものとして，①は石が死に，②は石が生きているということを意味します。したがって，黒から見れば，黒先黒生のときには，①は勝，黒先白死のときは，②が負ということになります。白から見た場合は当然この逆です。これをまとめたのが**表10-2**です。

図10-3のフローチャート（次ページ）では，このように①や②で終了するかどうか判定し，その場合は上のような結果を変数Vの値として返しています。

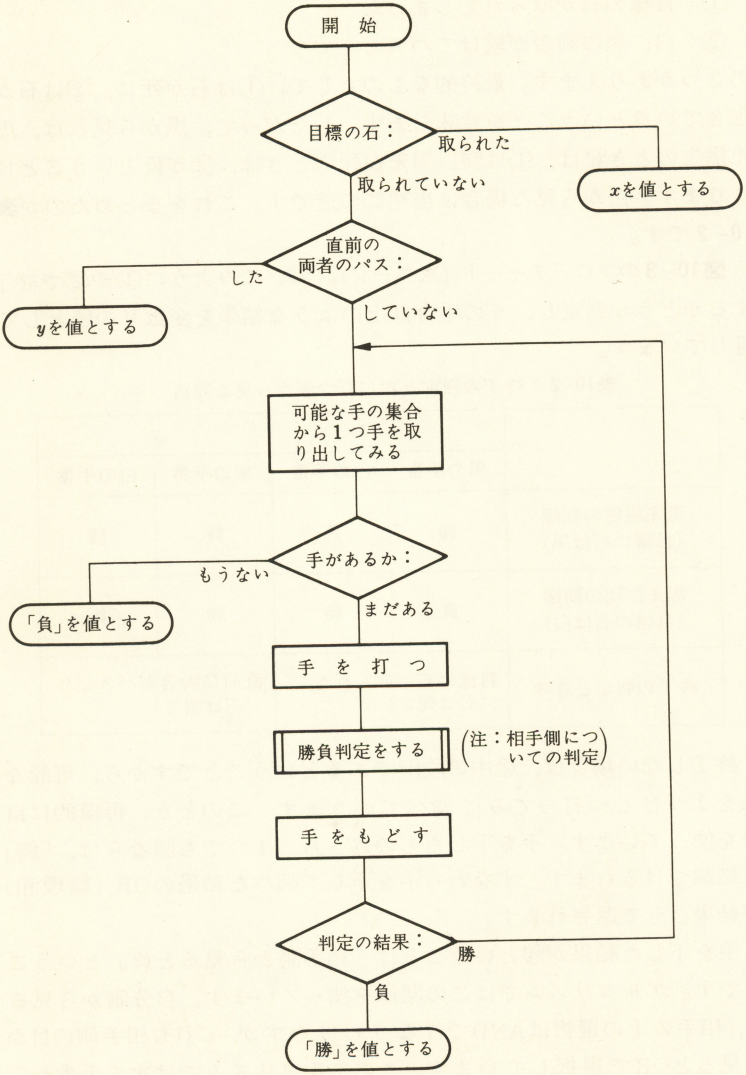
表10-2　終了の判定と次に打つ側から見た勝負

	x		y	
	黒の手番	白の手番	黒の手番	白の手番
黒先黒生の問題 （目標の石は黒）	勝	負	負	勝
黒先白死の問題 （目標の石は白）	負	勝	勝	負
終了の判定と意味	目標の石が取られた （石は死に）		直前に両者がパスした （石は生き）	

終了しない場合は，途中の段階であるということですから，可能な手を1つひとつ打ってみて調べていきます。このとき，再帰的に自分を使っています。手を下したもののうち，1つでも勝ならば，「勝」と結論づけられます。すなわち手を下して調べた結果のOR（論理和）が結果として返されます。

手を下した結果が勝ということは「相手側から見ると負」ということです。アルゴリズムではこの関係を使っています。自分側から見ると，相手の手の選択はANDで行なうわけですが，これも相手側の目から見るとORで選択しています。このアルゴリズムでは次の手番から見て考えていますから，全部ORで処理できることになります。

結局，可能な手を1つずつ打って調べていき，相手の負ならば，こ



(x，yは表10－2に与える)

図10－3　勝負判定のアルゴリズム

ちらの勝，全部調べきってもう手がなくなれば，こちら側の負になります。

●詰碁を解くプログラム

プログラムを**図10-4**に示します。また，結果は**図10-5**です。これは**図10-6**の問題を解き**図10-7**のようなゲーム木を調べたことを表わしています。

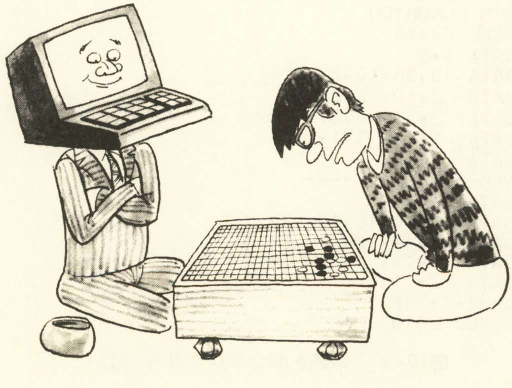
プログラムの主部は，可能な手全部について，勝負判定のサブルーチンを呼び，もし1つの手が勝ならばその手を，

　　コタエ　ハ　X　ノ　Y　デス。

というように出します。勝がなければ，

　　コタエ　ハ　アリマセン。

と出力します。



```
0010 PRINT"--- ツメゴ ---"
0020 DIM B(19,19),X9(4),Y9(4)
0030 REM TORISTK YOMISTK
0040 DIM S(1000),S1(1000),C$(3)
0050 REM UTERU BASHO
0060 DIM X7(200),Y7(200)
0070 REM TE NO STACK NO KATA
0080 REM X1,Y1,...XN,YN,          X,Y, -99
0090 REM KONO ISHINO TSUZUKI TORU, UCHI,MARK
0100 READ C$(1),C$(2),C$(3)
0110 DATA " ."," *"," O"
0120 REM 4-HOUKOU
0130 FOR D=1 TO 4
0140    READ X9(D),Y9(D)
0150    NEXT D
0160 DATA 0,1,0,-1,1,0,-1,0
0170 REM START
0180 FOR T=1 TO 10
0190    P=0
0200    P1=0
0210    P1=P1+1
0220    S1(P1)=-99
0230    N7=0
0240    READ X3,Y3
0250    FOR X=1 TO X3
0260       FOR Y=1 TO Y3
0270          READ B(X,Y)
0280          IF B(X,Y)<10 THEN 330
0290          B(X,Y)=B(X,Y)-10
0300          N7=N7+1
0310          X7(N7)=X
0320          Y7(N7)=Y
0330          NEXT Y
0340       NEXT X
0350    N7=N7+1
0360    X7(N7)=0
0370    Y7(N7)=0
0380    REM MOKUHYOU
0390    READ X8,Y8
0400    DATA 4,5
0410    DATA 10,10,10,2,1
0420    DATA  2,10, 2,2,1
0430    DATA  2, 2, 2,2,1
0440    DATA  1, 1, 1,1,1
0450    DATA 2,1
0460    REM--------------
0470    DATA 3,6
0480    DATA 10,10,10,10, 2, 1
0490    DATA  2, 2, 2, 2, 2, 1
0500    DATA  1, 1, 1, 1, 1, 1
0510    DATA  2, 1
0520    L=B(X8,Y8)
0530    GOSUB 2010
```

図10-4　詰碁を解くプログラム　①

```
0540    PRINT "<START>"
0550    I=1
0560    REM KI O HIROGERU
0570    FOR N6=1 TO N7
0580       X=X7(N6)
0590       Y=Y7(N6)
0600       GOSUB 750
0610       IF U=0 THEN 690
0620       I=3-I
0630       GOSUB 2100
0640       GOSUB 1610
0650       I=3-I
0660       IF V=1 THEN 690
0670       PRINT "ｺﾀｴ ﾊ ";X7(N6);" ﾉ";Y7(N6);" ﾃﾞｽ."
0680       GOTO 710
0690       NEXT N6
0700    PRINT "ｺﾀｴ ﾊ ｱﾘﾏｾﾝ."
0710    NEXT T
0720 STOP
0730 REM-----------------------------------
0740 REM SUBR X,Y NI I O UTERUKA?
0750 REM KANOU NARA UTSU
0760 REM PASS
0770 IF X<>0 THEN 800
0780 IF Y<>0 THEN 800
0790 GOTO 1220
0800 U=0
0810 IF B(X,Y)<>0 GOTO 1210
0820 B(X,Y)=I
0830 T1=0
0840 FOR D=1 TO 4
0850    X=X+X9(D)
0860    Y=Y+Y9(D)
0870    P=P+1
0880    S(P)=D
0890    IF X<1 THEN 1080
0900    IF Y<1 THEN 1080
0910    IF X>19 THEN 1080
0920    IF Y>19 THEN 1080
0930    IF B(X,Y)+I<>3 THEN 1080
0940    I=3-I
0950    GOSUB 1340
0960    R1=-I
0970    R2=I
0980    IF T=0 THEN 1060
0990    REM TORERU
1000    P1=P1+1
1010    S1(P1)=X
1020    P1=P1+1
1030    S1(P1)=Y
1040    R2=0
1050    T1=T1+1
```

図10-4　詰碁を解くプログラム ②

```
1060     GOSUB 1810
1070     I=3-I
1080     D=S(P)
1090     P=P-1
1100     Y=Y-Y9(D)
1110     X=X-X9(D)
1120     NEXT D
1130 IF T1>0 THEN 1220
1140 GOSUB 1340
1150 R1=-I
1160 R2=I
1170 GOSUB 1810
1180 U=1-T
1190 IF U=1 THEN 1220
1200 B(X,Y)=0
1210 RETURN
1220 U=1
1230 P1=P1+1
1240 S1(P1)=X
1250 P1=P1+1
1260 S1(P1)=Y
1270 P1=P1+1
1280 S1(P1)=N6
1290 P1=P1+1
1300 S1(P1)=-99
1310 RETURN
1320 REM---------------------------------
1330 REM SUBR TORERUKA?
1340 IF X<1 GOTO 1570
1350 IF Y<1 GOTO 1570
1360 IF X>19 GOTO 1570
1370 IF Y>19 GOTO 1570
1380 IF B(X,Y)<0 GOTO 1570
1390 IF B(X,Y)+I=3 GOTO 1570
1400 IF B(X,Y)=I GOTO 1430
1410 T=0
1420 RETURN
1430 B(X,Y)=-B(X,Y)
1440 FOR D=1 TO 4
1450     X=X+X9(D)
1460     Y=Y+Y9(D)
1470     P=P+1
1480     S(P)=D
1490     GOSUB 1340
1500     D=S(P)
1510     P=P-1
1520     Y=Y-Y9(D)
1530     X=X-X9(D)
1540     IF T=1 GOTO 1560
1550     RETURN
1560     NEXT D
1570 T=1
1580 RETURN
```

図10-4 詰碁を解くプログラム ③

```
1590 REM--------------------------------
1600 REM SUBR TE O MODOSU
1610 P1=P1-1
1620 N6=S1(P1)
1630 P1=P1-1
1640 Y2=S1(P1)
1650 P1=P1-1
1660 X2=S1(P1)
1670 P1=P1-1
1680 IF S1(P1)=-99 THEN 1770
1690 R1=0
1700 R2=I
1710 Y=S1(P1)
1720 P1=P1-1
1730 X=S1(P1)
1740 P1=P1-1
1750 GOSUB 1810
1760 GOTO 1680
1770 IF X2<1 THEN 1790
1780 B(X2,Y2)=0
1790 RETURN
1800 REM--------------------------------
1810 REM SUBR ZUKEI-NURIKAE:R1>R2 AT(X,Y)
1820 IF X<1 THEN 1990
1830 IF Y<1 GOTO 1990
1840 IF X>19 GOTO 1990
1850 IF Y>19 GOTO 1990
1860 IF B(X,Y)<>R1 THEN 1990
1870 B(X,Y)=R2
1880 FOR D=1 TO 4
1890    X=X+X9(D)
1900    Y=Y+Y9(D)
1910    P=P+1
1920    S(P)=D
1930    GOSUB 1810
1940    D=S(P)
1950    P=P-1
1960    Y=Y-Y9(D)
1970    X=X-X9(D)
1980    NEXT D
1990 RETURN
2000 REM--------------------------------
2010 REM SUBR PRINT
2020 FOR X2=1 TO X3+1
2030    FOR Y2=1 TO Y3+1
2040       PRINT C$(B(X2,Y2)+1);
2050       NEXT Y2
2060    PRINT
2070    NEXT X2
2080 RETURN
2090 REM--------------------------------
2100 REM SUBR VAL(STATE)
2110 FOR P2=2 TO P1
```

図10-4　詰碁を解くプログラム　④

```
2120     IF S1(P2)<>-99 THEN 2140
2130     PRINT "->";
2140     NEXT P2
2150 PRINT C$(3-I+1);TAB(30);C$(3-I+1);X;Y
2160 REM 2KAI PASS
2170 IF P1<9 THEN 2220
2180 IF S1(P1-3)<>0 THEN 2220
2190 IF S1(P1-7)<>0 THEN 2220
2200 V=(3-2*L)*(3-2*I)
2210 RETURN
2220 REM MOKUHYOU O TOTTA
2230 IF B(X8,Y8)<>0 THEN 2260
2240 V=-(3-2*L)*(3-2*I)
2250 RETURN
2260 REM KI O HIROGERU
2270 FOR N6=1 TO N7
2280     X=X7(N6)
2290     Y=Y7(N6)
2300     GOSUB 750
2310     IF U=0 THEN 2390
2320     I=3-I
2330     GOSUB 2100
2340     GOSUB 1610
2350     I=3-I
2360     IF V=1 THEN 2390
2370     V=1
2380     RETURN
2390     NEXT N6
2400 V=-1
2410 RETURN
2420 END
```

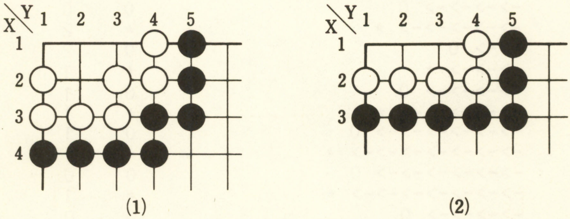
図10-4　詰碁を解くプログラム　⑤

```
--- ツ メ ゴ ---
 . . . 0 * .
 0 . 0 0 * .
 0 0 0 0 * .
 * * * * * .
 . . . . . .
<START>
-> *                                  *   1   1
->-> 0                                0   1   2
->->-> *                              *   0   0
->->->-> 0                            0   1   1
->->->->-> *                          *   0   0
->->->->->-> 0                        0   1   3
->->->->->->-> *                      *   2   2
->->->->->-> 0                        0   2   2
->->->->->->-> *                      *   1   3
->->->->->-> 0                        0   0   0
-> *                                  *   1   2
->-> 0                                0   1   1
->->-> *                              *   1   3
->->->-> 0                            0   2   2
->->->->-> *                          *   1   2
->->->->->-> 0                        0   1   3
->->->->->->-> *                      *   1   2
->->->->->-> 0                        0   0   0
->->->->->->-> *                      *   1   3
->->->-> 0                            0   0   0
->->->->-> *                          *   2   2
->-> 0                                0   1   3
->->-> *                              *   1   1
->->->-> 0                            0   2   2
->->->->-> *                          *   1   1
->->->->->-> 0                        0   1   2
->->->->->->-> *                      *   1   1
->->->->->-> 0                        0   0   0
->->->->->->-> *                      *   1   2
->->->-> 0                            0   0   0
->->->->-> *                          *   2   2
->-> 0                                0   2   2
->->-> *                              *   1   1
->->->-> 0                            0   1   3
->->->->-> *                          *   1   1
->->->->->-> 0                        0   1   2
->->->->->->-> *                      *   1   1
->->->->->-> 0                        0   0   0
->->->->->->-> *                      *   1   2
->->->-> 0                            0   0   0
->->->->-> *                          *   1   3
->-> 0                                0   0   0
->->-> *                              *   1   1
->->->-> 0                            0   1   3
->->->->-> *                          *   2   2
->->->-> 0                            0   2   2
```

図10-5 結 果 ①

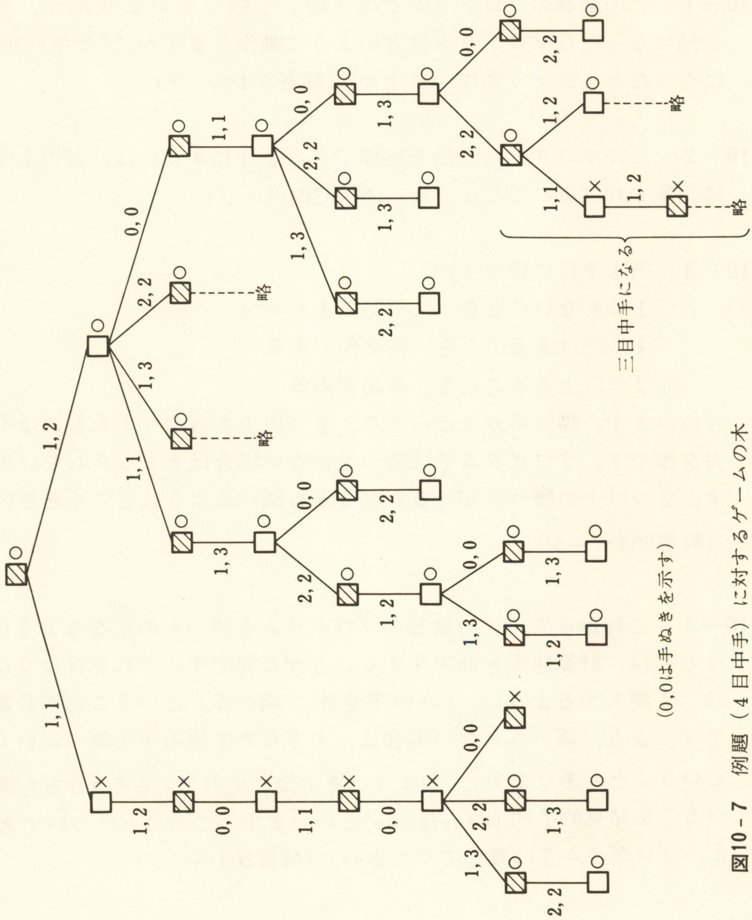
```
->->->->-> *                   *    1     3
->->->-> 0                     0    0     0
->->->->-> *                   *    1     3
->->->->->-> 0                 0    2     2
->->->->->->-> *               *    1     1
->->->->->->->-> 0             0    1     2
->->->->->->->->-> *           *    0     0
->->->->->->->->->-> 0         0    1     1
->->->->->->->->->->-> *       *    1     3
->->->->->->->->->-> 0         0    1     3
->->->->->->->->->->-> *       *    1     1
->->->->->->->->->-> 0         0    0     0
->->->->->->-> *               *    1     2
->->->->->->->-> 0             0    1     1
->->->->->->->->-> *           *    1     3
->->->->->->->-> 0             0    1     3
->->->->->->->->-> *           *    1     1
->->->->->->->-> 0             0    0     0
->->->->->->->->-> *           *    1     1
->->->->->->->->->-> 0         0    1     3
->->->->->->->->->->-> *       *    1     1
->->->->->->->->->->->-> 0     0    1     2
->->->->->->->->->->->->-> *   *    1     1
->->->->->->->->->->->-> 0     0    0     0
->->->->->->->->->->->->-> *   *    1     2
->->->->->->->->->-> 0         0    0     0
->->->->->->->->->->-> *       *    1     3
->->->->->-> 0                 0    0     0
->->->->->->-> *               *    2     2
コタエ ハ     1  ノ    2  デ゛ス.
```

図10-5　結　果　②



(1)　　　　(2)

図10-6　例　題



(0,0は手ぬきを示す)

図10-7　例題（4目中手）に対するゲームの木

課　題

10—1　この詰碁のプログラムでは「劫」を扱っていませんが，もし劫になったらプログラムはどのように動作しますか。「セキ」の形になったときはどう動作しますか。(解答204ページ)

10—2　このプログラムで劫を処理できるようにするには，どのように改善すればよいでしょうか。(解答204ページ)

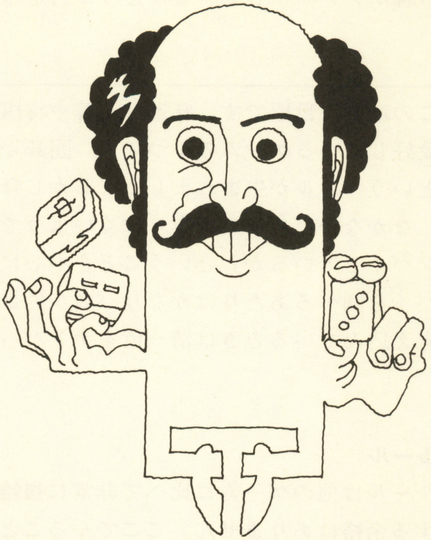
10—3　第1手目に勝つ手が，

1つもないことを　不詰である
1つだけあることを　完全作である
2つ以上あることを　余詰がある

といいます。勝つ手がなかったり，2つ以上あったりするものは不完全作です。プログラムでは勝つ手がない場合はチェックしています。2つ以上の勝つ手があるかどうかも調べるようにしてください。(解答204ページ)

10—4　これから先，この詰碁のプログラムを使いものになるようにするには，計算速度を効率化することが必要です。これを行なうには，人間もするように，「いい手を先に調べる」ということが必要です。また，調べきれない場合は，あきらめて他の手を調べにいくということもあります。このようにその場に応じてよさそうな方を調べるのを発見的手法 (heuristics)といいます。このことについて考え，プログラムを改善してください。(解答204ページ)

11★麻雀のパズル



【問　題】

13個の牌(パイ)がテンパイしているかどうか判定し，上がり牌を調べる。

これがこの章での目標です。麻雀は囲碁や将棋とならんで日本人がもっとも愛好しているゲームの1つです。囲碁からは詰碁，将棋からは詰将棋というパズルが生まれました。しかし麻雀は確率的要素が強いためか，なかなかパズルにはなりにくいようです。ここでは「メンゼンでテンパイしているか」ということを中心に考えてみます。清一色(チンイーソー)のテンパイを調べるあたりはかなりパズル的といってよいでしょう。初心者が考え込んでいるときは清一色をテンパっていることが多いですね。

●麻雀のルール

麻雀のルールは他のゲームに比べて非常に複雑ですから，ここで全部を説明する余裕はありません。ここでやることを中心に少し説明します。

上がりの形は14枚で，普通2つ組が1つと3つ組を4つからできています。この上がり形になることを4人が競争するわけです。

2つ組は，同じ牌2個のことで**頭**といいます。3つ組は**対子**(トイツ)と**順子**(シュンツ)の2種あります。対子は同じ牌3個のことです。順子は数牌が2, 3, 4, とか7, 8, 9のように順に並んだものです。

テンパイとは上がりの形に1枚牌が足りない形のことです。ですから個数は13個しかありません。テンパイの形に加えると上がりの形になるような1枚の牌を**待ち牌**といいます。待ち牌がn種類あれば***n*面待ち**といいます。

清一色は数牌のうち1種類からだけ構成されている形をいいます。数牌はおのおの1から9まであります。

●目標

清一色のテンパイを調べることをまず考えます。一般の場合を調べるのに対して，問題は少しやさしくなります。ところが人間にとっては逆にわかりにくくなります。牌が重なって，待ちの数が多くなるからです。ここにパズル的な要素があります。

清一色に限れば，牌は1から9までの数として簡単に表わせます。13個の1から9の間の数が，システムへの入力になります。出力は待ち牌と上がりの形のリストです。

例をあげてみましょう。たとえば，

1 1 1 2 2 2 3 3 3 4 4 4 5

の13個の牌があったとします。これはテンパイ形です。なぜかというと，5を加えると，

1 2 3	1 2 3	1 2 3	4 4 4	5 5
順子	順子	順子	対子	頭

となり，上がり形が完成します。また4を加えると，

1 1 1	2 2 2	3 4 5	4 4 4	3 3
対子	対子	順子	対子	頭

となり，完成します。これ以外にも別の牌でいろいろな上がり形ができます。

こういうものを全部調べてみるプログラムを作ってみることにします。

●計算のアルゴリズム

調べ方のあらましは，どのような方法によっても次のようになるでしょう。

(1)　待ち牌（1個）を調べる牌（13個）に付け加えてみる。待ち牌を1から9まで変えておのおのの場合を全部調べる

(2)　頭となる同じ牌2個を取り除いてみる。頭を1から9まで2枚以上あるものについて変えながら，おのおのの場合を全部調べる

(3)　最小数の牌を含む対子があるかどうか調べる。あれば取り除く

(4)　なければ，最小数の牌を含む順子があるかどうか調べる。あれ

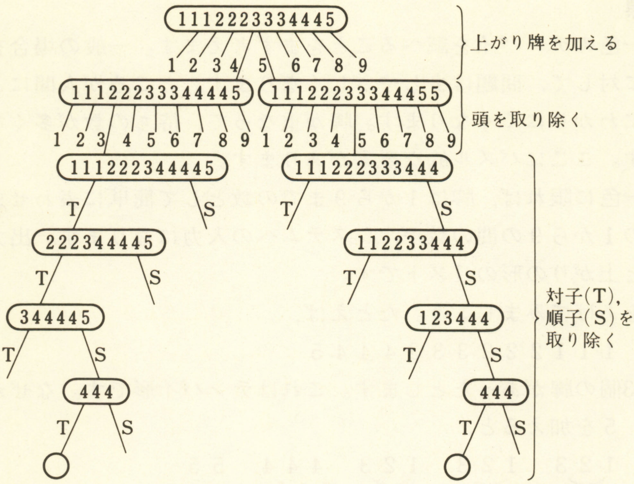


図11-1　テンパイを調査する探索木

ば取り除く

(5)　なければテンパイしていない

(6)　(3)と(4)を4回繰り返して成功すればテンパイしている

この考え方を使って探索木を描いてみます（**図11-1**）。探索の例として前に述べた，

1 1 1 2 2 2 3 3 3 4 4 4 5

の場合を調べてみます。各状態に対してその場にある牌を書いてあります。多くの部分を省略してありますが，これで探索の方法が理解しやすくなったと思います。

この探索法によって，全部の上がり方をまとめて扱うことができます。つまり単騎(タンキ)待ち，両面(リャンメン)待ち，辺張(ペンチャン)，間張(カンチャン)，双碰(シャンポン)にいちいち区別しなくてもいいのです。

探索の方法をフローチャートに書いてみましょう。まず，上がり牌を加えることと，頭を取り除くことはFOR～NEXTループを使うことにします（**図11-2**）。残った12枚が4組の完成面子になっているかを調べるところが少し複雑になります（**図11-3**）。細部は省略してあり

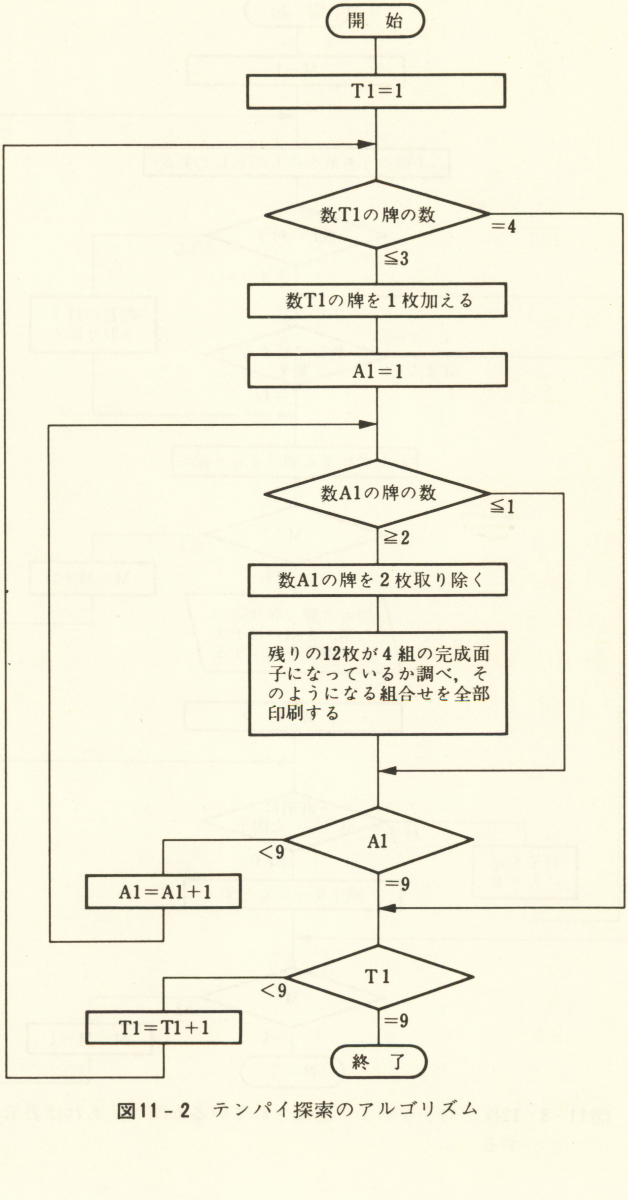


図11－2 テンパイ探索のアルゴリズム

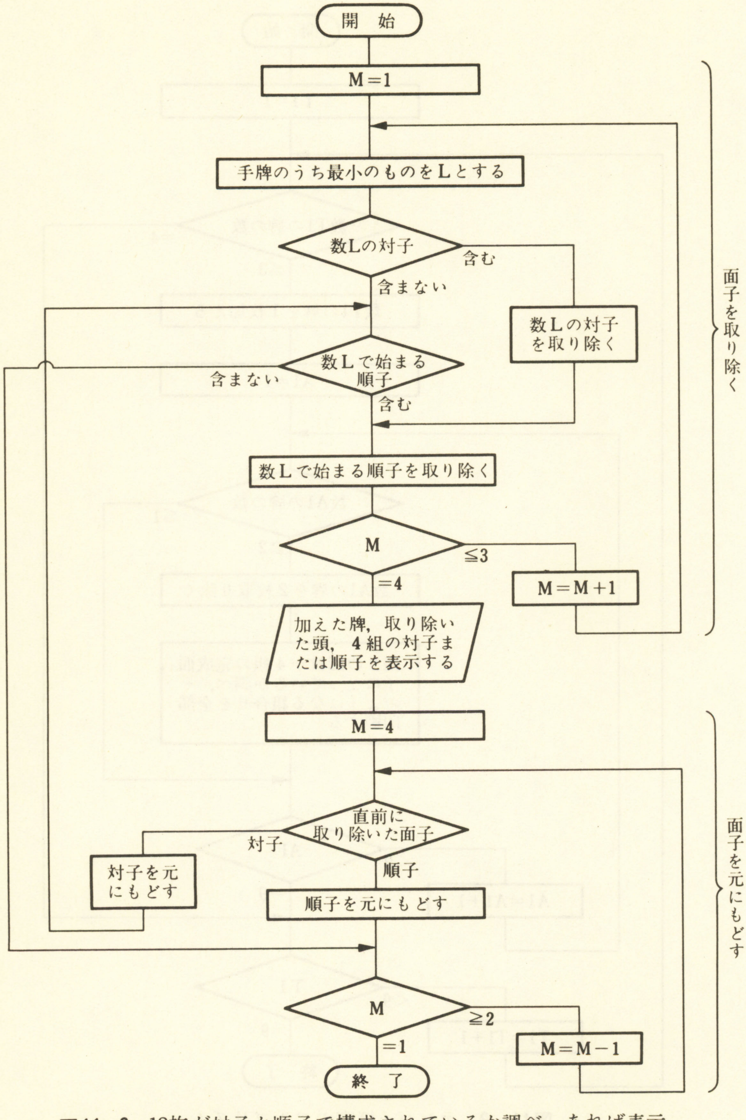


図11-3 12枚が対子か順子で構成されているか調べ，あれば表示する

```
0010 PRINT "ﾒﾝｾﾞﾝ ﾁﾝｲﾁ ﾉ ﾃﾝﾊﾟｲ ｼﾗﾍﾞ"
0020 DIM K(9),L1(4),S(4)
0030 FOR T=1 TO 100
0040    FOR J=1 TO 9
0050       K(J)=0
0060       NEXT J
0070    PRINT "ﾓﾝﾀﾞｲ :<";
0080    P1=0
0090    FOR Q=1 TO 13
0100       IF P1<100 THEN 130
0110       PRINT P1;
0120       P1=0
0130       READ P
0140       P1=10*P1+P
0150       K(P)=K(P)+1
0160       NEXT Q
0170    PRINT P1;">"
0180    DATA 1,1,1,2,3,4,5,6,7,8,9,9,9
0190    DATA 1,1,1,2,2,2,3,3,3,4,4,4,5
0200    DATA 1,2,3,4,5,6,7,8,9,4,4,4,5
0210    REM---TSUMO
0220    FOR T1=1 TO 9
0230       IF K(T1)=4 THEN 760
0240       K(T1)=K(T1)+1
0250       REM---ATAMA O TORU
0260       FOR A1=1 TO 9
0270          IF K(A1)<2 THEN 740
0280          K(A1)=K(A1)-2
0290          L=1
0300          REM---TORISARU
0310          FOR M=1 TO 4
0320             IF K(L)<>0 THEN 350
0330             L=L+1
0340             GOTO 320
0350             IF K(L)<3 THEN 390
0360             IF M=1 THEN 490
0370             IF S(M-1)=0 THEN 490
0380             IF L1(M-1)<>L THEN 490
0390             REM---SHUNTSU TEST
0400             IF L>7 THEN 720
0410             IF K(L+1)=0 THEN 720
0420             IF K(L+2)=0 THEN 720
0430             REM---SHUNTSU
0440             K(L)=K(L)-1
0450             K(L+1)=K(L+1)-1
0460             K(L+2)=K(L+2)-1
0470             S(M)=1
0480             GOTO 520
0490             REM---TOITSU
0500             K(L)=K(L)-3
0510             S(M)=0
0520             L1(M)=L
0530             NEXT M
```

図11-4 清一色のテンパイを調べるプログラム ①

```
0540           REM---AGARI
0550           PRINT T1;" ﾏﾁ---";11*A1;
0560           FOR M=1 TO 4
0570              PRINT 111*L1(M)+12*S(M);
0580              NEXT M
0590           PRINT
0600           REM---MENTSU O MODOSU
0610           FOR M=4 TO 1 STEP -1
0620              IF S(M)=1 THEN 670
0630              REM---TOITSU
0640              L=L1(M)
0650              K(L)=K(L)+3
0660              GOTO 390
0670              REM---SHUNTSU
0680              L=L1(M)
0690              K(L)=K(L)+1
0700              K(L+1)=K(L+1)+1
0710              K(L+2)=K(L+2)+1
0720              NEXT M
0730           K(A1)=K(A1)+2
0740           NEXT A1
0750        K(T1)=K(T1)-1
0760        NEXT T1
0770     NEXT T
0780 END
```

図11-4 清一色のテンパイを調べる
プログラム ②

ますが，探索の様子が理解できると思います。流れは，主として面子を取り除く部分と，面子を元にもどす部分に分かれ，その間を行ったり来たりします。今までと同じようにこれも縦型探索ですが，違う形に設計しました。

●テンパイを調べるプログラム

プログラムを**図11-4**に示します。手の形は牌に対応する13個の1から9までの数値で与えます。これを牌の出現個数の配列Kに変換します。つまり，牌のデータが，

1，1，1，2，3，4，5，6，7，8，9，9，9

であれば，K(1)，K(2)，……，K(9)は

3，1，1，1，1，1，1，1，3

となります。処理の段階ではこのデータ構造の方が元の形ですますよりも便利です。この場合もデータ構造の選択がキーポイントだと思い

ます。

縦型探索に必要なスタックが，プログラムの中に見あたらないと思われるかもしれません。ここでは今までの方法と違い，スタックそのものは使いませんが，配列L1とSがその役を果たしています。L1(M)はM番目の面子（対子か順子）の最初の牌の位置を表わしています。S(M)はM番目の面子が対子か順子かを表わします。対子なら0，順子なら1です。

結果を**図11-5**に示します。同じ待ちでも違う形になるものは全部表示しています。たとえば，1 1 1 2 2 2 3 3 3 4 4 5に3がきた場合には，

1 1　1 2 3　2 3 4　2 3 4　3 4 5

と，

4 4　1 1 1　2 2 2　3 3 3　3 4 5

```
ﾒﾝｾﾞﾝ ﾁﾝｲﾁ ﾉ ﾃﾝﾊﾟｲ ｼﾗﾍﾞ
ﾓﾝﾀﾞｲ :<  111    234    567    899      9 >
      1  ﾏﾁ---    99    111    123    456    789
      2  ﾏﾁ---    22    111    345    678    999
      3  ﾏﾁ---    11    123    345    678    999
      4  ﾏﾁ---    99    111    234    456    789
      5  ﾏﾁ---    55    111    234    678    999
      6  ﾏﾁ---    11    123    456    678    999
      7  ﾏﾁ---    99    111    234    567    789
      8  ﾏﾁ---    88    111    234    567    999
      9  ﾏﾁ---    11    123    456    789    999
ﾓﾝﾀﾞｲ :<  111    222    333    444      5 >
      2  ﾏﾁ---    22    111    234    234    345
      3  ﾏﾁ---    11    123    234    234    345
      3  ﾏﾁ---    44    111    222    333    345
      3  ﾏﾁ---    44    123    123    123    345
      4  ﾏﾁ---    33    111    222    345    444
      5  ﾏﾁ---    22    111    234    345    345
      5  ﾏﾁ---    55    111    222    333    444
      5  ﾏﾁ---    55    111    234    234    234
      5  ﾏﾁ---    55    123    123    123    444
      6  ﾏﾁ---    11    123    234    234    456
      6  ﾏﾁ---    44    111    222    333    456
      6  ﾏﾁ---    44    123    123    123    456
ﾓﾝﾀﾞｲ :<  123    456    789    444      5 >
      3  ﾏﾁ---    44    123    345    456    789
      5  ﾏﾁ---    55    123    444    456    789
      6  ﾏﾁ---    44    123    456    456    789
```

図11-5　テンパイの結果

の２通りがあります。実際，役を計算すれば，おのおの点が違ってきます。当然高い点の方を選びます。そのためには両方を調べるように作っておかなければなりません。上の場合では前のものは一ペイコー，平和の役ですが，後のものは自模(ツモ)ったときに三暗刻になるだけです。雀球（パチンコ型の麻雀遊戯機）で点数計算を自動的にしてくれるのが普及していますが，こういう点で手を抜いているのがあるようです。

●答の表示

答の表示のところでは，少しこったことをしています。

PRINT　111＊L1(M)＋12＊S(M)

をよく考えてください。うまく対子や順子が表示されます。

高級なBASICがあるなら，PRINT　USING文を使った方が正統的でしょう。

●一般の場合のテンパイを調べる

清一色（同一種の数牌からできている場合）を考えてきましたが，さらに一般的な場合に拡張してみましょう。

清一色の場合は，牌の種類は９種でしたが，麻雀には34種あります。また，このうち字牌（東，南，西，北，白，發，中）は順子になりません。また数牌は３組（筒子，索子，萬子）あり，おのおのその中でだけ順子を構成します。このことをプログラムの中で表現してやればよいのです。

１筒，２筒，……，９筒；
１索，２索，……，９索；
１萬，２萬，……，９萬；
東，南，西，北，白，發，中

におのおの，１，……，９；10，……，18；19，……，27；28，……，34を対応させることにしましょう。そうして，配列Ｆの要素F(1)，……，F(34)に，

0，0，0，0，0，0，0，1，1，
0，0，0，0，0，0，0，1，1，

0, 0, 0, 0, 0, 0, 0, 1, 1,

1, 1, 1, 1, 1, 1, 1

を入れておきます。すると,

F(L)＝0なら牌Lから始まる順子は作れる
F(L)＝1なら牌Lから始まる順子は作れない

ということになります。これを使って,

IF　L＞7　THEN〜

の行を

IF　F(L)＝1　THEN〜

で取り換えれば，一般の場合のプログラムを実現できます。後は牌の種類が9であるところを34に取り換える程度ですみます。

ここでも，**意味をデータ（配列）で表わす**ことによってうまくいくのがわかると思います。IF文やGOTO文でしようとすると面倒になります。

課　題

11—1　上に述べた一般の場合のテンパイを調べるプログラムを作ってください。(解答205ページ)

11—2　テンパイの探索を効率化してください。

方法としてはいろいろ考えられます。たとえば,

(1)　1枚牌を加えるときに，その牌のまわりに牌が存在しなければテンパイしている可能性がない。したがってその場合は探索しなくてよい。

(2)　頭を除いて12枚になったとき，残りは3の倍数個の牌のかたまりにわかれていなければならない。たとえば,

1 1 | 3 3 3 4 4 5 5 | 7 7 8 9 9
A | A | B

ではかたまりAやBの牌の数が3の倍数でないのでテンパイしていない。こういうものも探索しなくてよい。

(3)　頭を除いて12枚になったとき(清一色の場合)，Kのモーメント,

$$1 \cdot K(1) + 2 \cdot K(2) + \cdots\cdots + 9 \cdot K(9)$$

が3の倍数でなければ，テンパイしていないので探索しなくてよい。

(4) 探索の範囲を34牌全部にするのではなく，持牌のまわりに限定する。(解答205ページ)

11―3 一シャンテン（テンパイの一歩手前）かどうかを調べるようなものを作ってください。(解答205ページ)

11―4 上がり形に対して役と点を計算するようなものを作ってください。(解答205ページ)

11―5 データの入出力を数字ではなく，牌の名前でするようにプログラムを変えてください。(解答206ページ)

11―6 頭と4面子の一般の上がり形の他に2つの上がり形があります。すなわち，

(1) 七対子：7個の頭でできているもの。各頭は違う牌である。

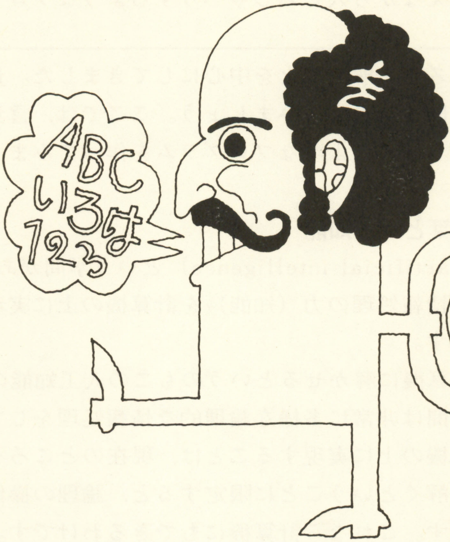
(2) 国士無双：幺九牌(ヤオチュー)（1と9の数牌）と字牌からできている異なる13牌プラス1牌

これらに関してもテンパイを調べるようにしてください。(解答206ページ)

11―7 待ち牌の数（牌の種類の数ではない）も出力するようにしてください。実戦ではこれで上がりやすさをある程度評価します。(解答206ページ)

11―8 ランダムに牌を選び出し，どの程度の確率でテンパイしているか調べてください。(解答207ページ)

12★しゃべる機械



【問　題】

言葉を覚えながら人とおしゃべりするようなプログラムを作る。

今まではパズルを解くことを中心にしてきました。最後にパズルから少し離れた問題を考えてみましょう。ここでは，言葉を覚えながら人とおしゃべりをするようなプログラムを作ってみます。

●パズルと言葉と人工知能

人工知能（artificial intelligence）という学問があります。これは人間の高度な情報処理の力（知能）を計算機の上に実現することを目標としています。

パズルを計算機に解かせるというのもこの人工知能の一部であるといえます。人間は非常に多様な論理的な情報処理をしていますが，これを全部計算機の上に実現することは，現在のところ不可能です。1つのパズルを解くということに限定すると，論理の操作を枠にはめることができます。これなら計算機にもできるわけです。このようにパズルは人工知能の分野において，ひとつの枠を与えるものであり，大きな役割を持っています。

一方，人間のように言葉を言ったり，理解したりするシステムの研究も人工知能のひとつの学問であり，多くの研究者が研究を進めています。これはパズルを解くのとは性格が違い，非論理的な動作であるという面がありますが，人間の情報処理能力の実現という点では同じです。

このようにパズルと言葉の処理の問題は，結びついています。

またここでは学習ということを考えてみます。学習とは環境からの情報の収集である，と定義できます。このことから考えてもパズルと関連してきます。こう考えてしゃべるプログラムの問題を取り上げてみたわけです。

●おしゃべりの例

まず**図12－1**を見てください。これはここで作ったプログラムとの対話の例です。Cが計算機側が出したもので，Mが人の打鍵したものです。全部読んでみてください。どんな感じがしますか。プログラムはそんなに難しいものではありません。しかし，この程度のもっともらしい答え方をすることができます。

計算機の応答はだいぶ変で，非常に奇妙な感じがすると思います。しかし，相手が機械であるという気はしにくいでしょう。人にたずねると，相手は狂人であるとか2歳ぐらいの幼児であるように感じる，といいます。

このようなことを行なうプログラムをどう作ればよいか考えてみます。

```
---talking machine---
ｵﾜﾙ ﾄｷ ﾊ # ｦ ｲﾚﾙ.
ｺﾝﾏ ﾊ ﾂｶﾜﾅｲﾃﾞ ｸﾀﾞｻｲ.
<START>
C:ｱﾅﾀ ﾉ ﾅﾏｴ ﾊ? ｿｳﾀﾞﾝ ｺﾞﾄ ﾊ?
M:anata wa dare?
C:anata no koto.
M:jibun no namae.
C:jibun?
M:shiranai no?
C:shiranai? sou desu.
M:komatta.
C:komatta?
M:jibun no namae kurai oboete oke.
C:oboete? komatta nee.
M:anata wa computer desu.
C:computer? jibun no namae kurai oboete oke.
M:jibun to iu no wa omae no koto da.
C:shiranai no?
M:shitteru yo.
C:shitteru? wakaranai no.
M:anata no atama ga wakaranai.
C:zenzen wakaranai no ne.
M:shumi wa.
C:shumi?
M:suki na koto.
C:suki? arale chan da to itta desho.
M:arale chan ga suki nano!
C:oboete kudasai.
```

図12－1 対話の一例 ①

```
M:daiji no koto da. oboeru yo.
C:oboeru? wakaranai no.
M:nani ga?
C:iinasai.
M:watashi ga desu ka?
C:watashi? shumi wa.
M:nai desu.
C:anata mo sou desu ne.
M:sou .
C:sou?
M:un.
C:un? sou desu.
M:yappari.
C:yappari?
M:kitto sou da to omotta.
C:omotta? jibun no namae.
M:ii namae.
C:namae? atama ga warukute komatte imasu.
M:watashi no namae o wasureta?
C:wasureta?
M:kitto  sou desho.
C:kitto?
M:un.
C:un? sou desu.
M:watashi wa tsuki yori no shisha desu.
C:nai desu.
M:chigau to iu no ka ne.
C:chigau?
M:utagau ka.
C:utagau? jibun no namae.
M:wakatta?
C:anata no atama ga wakaranai.
M:kamen o kabutteru kara ne.
C:kabutteru?
M:sou.
C:sou? utagau ka.
M:anata mo utagau?
C:wakatta?
M:wakaranai.
C:kamen o kabutteru kara ne.
M:baka ni naru to iu no?
C:baka? honto?
M:uso!!
C:uso? sou desu.
M:wadai o kaeyou.
C:kaeyou?
M:omoshiroi koto.
C:omoshiroi? chigau to iu no ka ne.
M:shumi ni tsuite.
C:tsuite? sou desu.
M:pazuru wa suki?
C:pazuru?
M:kuizu toka.
```

図12－1　対話の一例　②

```
C:Kuizu? nani ga?
M:1+1 wa nani?
C:iinasai.
M:watashi ga Kiite iru no da.
C:nai desu.
M:2 desho.
C:desho?
M:2 da.
C:da? sou desu.
M:1+1 wa?
C:1+1? watashi wa daijoubu.
M:Kotaete.
C:Kotaete?
M:1+1.
C:1+1? watashi wa daijoubu.
M:1+1 da.
C:1+1?
M:2 da.
C:da?
M:2!!
C:2? 2!!
M:mou ichido.
C:ichido?
M:1+1 wa?
C:2 da.
M:erai!!
C:erai? 2!!
M:atama ga yoku natta ne.
C:atama?
M:1+1 o oboeta.
C:oboeta? 2!!
M:sou da.
C:un.
M:computer mo oboeru.
C:atari!!!
M:nani oboeta?
C:sou da.
M:sore de.
C:sore? shumi wa.
M:shumi o oboeta?
C:oboeta?
M:wasureta?
C:Kitto  sou desho.
M:jaa watashi wa Kaeru.
C:nai desu.
M:sou iu na.
C:un.
```

図12-1 対話の一例 ③

●設計の方針

言葉をかわすシステムは，人工知能の１つの分野としていろいろな

研究が行なわれています。それらには，構文解析や意味解析など，複雑な処理がたいていあります。

ここではそのような複雑な処理をしません。複雑な処理をしないで，もっともらしい対話をさせようというのです。今までにそのようなものがあったかというと，ELIZAというシステムがありました。これは来訪者に対して，英語でカウンセリングをするようなプログラムです。

ELIZAは固定した応答しかせず，学習はしません。一方，ここでは，あらゆることを**学習**し，覚えていき，いろいろなことをしゃべるようになるようなプログラムを作ってみます。つまり，日本語で対話していると，だんだん日本語を学習していきます。英語で対話していると，英語をしゃべるようになるようにします。幼児が言葉を覚えるのと同じなわけで，何語でもいいのです。当然一方では，最初は何もしゃべれないということになります（**図12-2**）。

したがって，プログラムの中心の部分は，文章を覚えることと，適当な状況でそれを引き出して使うということになるでしょう。

```
---talking machine---
ｵﾜﾙ ﾄｷ ﾊ # ｦ ｲﾚﾙ.
ｺﾝﾏ ﾊ ﾂｶﾜﾅｲﾃﾞ ｸﾀﾞｻｲ.
<START>
C:ｱﾅﾀ ﾉ ﾅﾏｴ ﾊ? ｿｳﾀﾞﾝ ｺﾞﾄ ﾊ?
M:arale chan desu.
C:arale?
M:sou desu.
C:desu? arale chan desu.
M:oboete kudasai.
C:kudasai? sou desu.
M:atama ga warukute komatte imasu.
C:warukute?
M:sou desu.
C:desu?
M:anata mo sou desu ne.
C:anata?
M:anata no koto.
C:anata?
M:wakaranai no?
C:wakaranai?
M:zenzen wakaranai no ne.
C:wakaranai?
```

図12-2　最初の対話

●アルゴリズム

以下に述べることが，あのような応答をすることの種明しになります（種明しを読む前になぜできるのか少しは考えてください！)。

幼児の学習の本質が「まね」であるように，このアルゴリズムもまねに基づきます。この中身はかなり単純なものです。

今，計算機がP$という文を出力したとしましょう。そこへ人間がA$という応答をしたとします。学習とはこのP$とA$の関係を記憶することです。次に，たいていはしばらく時間がたちますが，人間が**P$に近い**ものを入力してきたとします。そこで記憶域を調べ，P$からA$を引き出し，それを人間に対して応答します。すなわち，計算機側は人間の行なったことをまねしているのです。逆に子供を観察してみると，これと同じことをしていることが実に多いことがわかります。

ここで“P$に近い”と書いたところが1つのキーポイントです。これがP$そのものであれば学習はほとんど進みません。つまりある程度幅を持った（似ているが少し違った）状況に反応しなければ，役に立ちません。実際に人間や動物でも，少し違うものに反応します。これを心理学の言葉で汎化（generalization）といいます。

ここで使っている方法は単純です。次のようにします。

①　まず人間の応答文の中から最も長い単語を取り出す。

②　その単語の左から4文字を取り出す。

こうして得られた文字列で文を代表するわけです。なぜこのようにするかというと，文の中で最も長い単語が最も重要である場合が多いから，①を行なうのです。また活用で語尾が変化したものも同じものとして見るために，②を行ないます。

こうして作られた4文字（以下）の列が同じであるような文同士が，“近い”ということにするのです。これがいわば「ミソ」のところで，これで結構うまくいくのです。

まねするものがないときの応答は，「おうむ返しに聞く」ということをします。これもまねの1つですが，さき程のまねに比べて，程度の低い反応です。しかしこれによって，人間側からなんらかの説明や対応が引き出されます。おうむ返しに言われると，人間はたいてい詳し

く説明しようとするのです。この人間の説明などが，まさに学習の対象になるわけです。

次にそのおうむ返しにしなければならなかった言語が出てきたときに，計算機はその詳しい説明をはじめるのです！

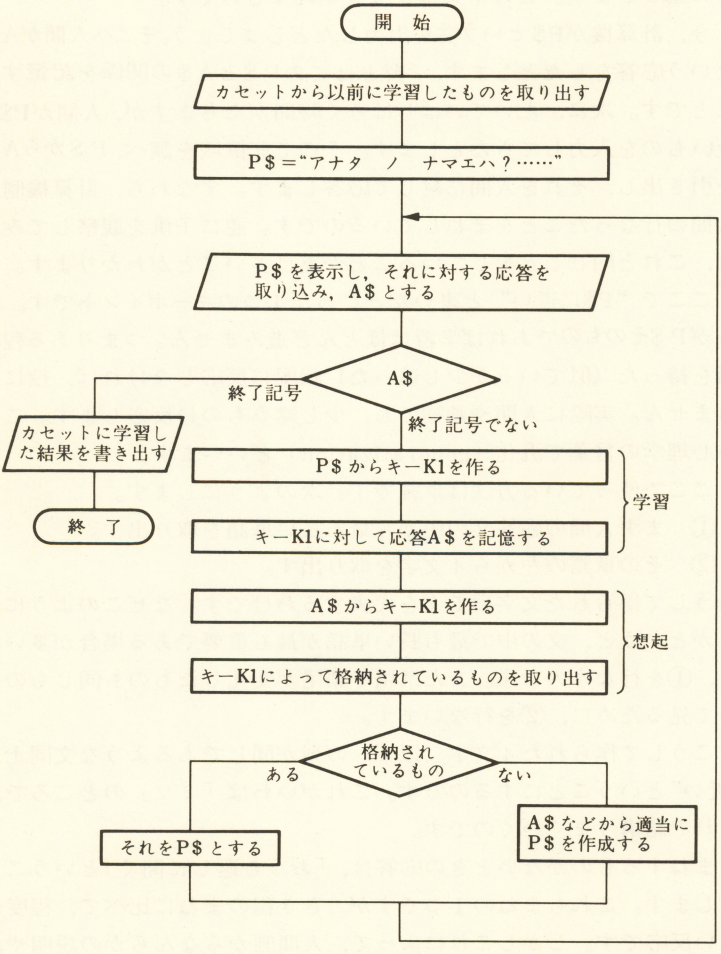


図12－3　しゃべる機械の概略のフローチャート

図12－3にアルゴリズムのフローチャートを示します。

●ハッシュ法

対話プログラムではハッシュ法（hash method）という探索法を使います。これを説明します。**図12－4**のような1万人の電話番号簿を考えてみます。名前から電話番号を知るにはどうしたらいいでしょうか。最も原始的な方法は，上から名前が求めるものであるかチェックしていく方法です。これを線形探索（linear search）といいます。これでは平均すると5000回も判定しなければなりません。これでは困るので，考えられたのがハッシュ法です。

ハッシュ法はハッシュ関数というものを使います。ハッシュ関数とは探索のキーを与えると，表での項目を格納する位置を出すようなものです。ハッシュ関数の性質としては，

①　値がなるべく散らばって一箇所に重なりにくいこと

②　計算が簡単なこと

を持っている必要があります。

上の例では名前がキーになります。ハッシュ関数をhで表わすと，項目（名前と電話番号）は表のh(名前)番目に格納されます。電話番号を見たいときにもh(名前)を調べ，その場所を見ます。項目があればその中の電話番号がわかります。

ハッシュで問題となるのは，異なるキーに対してハッシュ関数の値が同じになること，

1	カトウ　カズオ	03－456－7890
2	ヨシダ　ヨシオ	03－570－1982
3	イケダ　イクコ	0423－81－4221
⋮	⋮	
9999	アオキ　アヤコ	0429－43－2022
10000	オカド　オウイチ	044－61－6481

図12－4　電話番号簿

h(キー1)＝h(キー2)

です。これを**衝突**といいます。普通，衝突したら適当な方法でずらして格納したりします。ここでもそうすることにします。

ここではキーとなるのは計算機側の出した文を単純化したもの（もっとも長い単語の左から4文字）です。項目の中身にはキーのほかに，計算機の文に対する人間側の応答の文があります（その単純化したものも効率を考えて並べてあります）。ハッシュ関数としては，

h(x)＝〔xの長さ×5＋xの最も左の文字のコード
　　＋xの最も右の文字のコード＊100〕を
　　テーブルの大きさで割ったあまり

にしています。もちろん，違う形もありえます。

このようにハッシュ関数を使えば，衝突のことを考えても，数回のチェックで求めるものが得られます。ハッシュ法は，大きなソフトウエアではよく使われます。身につけてください。

●しゃべる機械のプログラム

プログラムを**図12-5**に示します。もっとも長い単語を取り出す部分を，サブルーチンにしてあります。

ここで使っているBASICは，基本BASICより少し大きいものを使っています（なるべく基本BASICに沿うようにしてありますが）。しかし広く使われているMicrosoft系BASICの上では，すべて動くはずです。そのようにしなければならなかったのは，

① 文字列の処理
② カセットへの入出力

の2つがあるからです。

①としては文字列のもっとも左（右）の文字を与える関数LEFT＄(RIGHT＄)，文字列の途中の文字をとり出す関数MID＄，文字からコードを与える関数ASC，文字列を並べる演算子“＋”を使っています。②としては，カセットへの入出力であるPRINT＃－1，INPUT＃－1を使いました。この部分を変えれば，他のBASICの上でも動かすことができるでしょう。

```
1010 PRINT "---TALKING MACHINE---"
1020 PRINT "ｵﾜﾙ ﾄｷ ﾊ # ｦ ｲﾚﾙ."
1030 PRINT "ｺﾝﾏ ﾊ ﾂｶﾜﾅｲﾃﾞ ｸﾀﾞｻｲ."
1040 PRINT "<START>"
1050 CLEAR 3000,59902!
1060 DIM K$(300),R$(300),R1$(300),D$(5)
1070 DEF FNK(H,N)=FNM(H+N*N)
1080 DEF FNM(M1)=M1-INT(M1/E)*E+1
1090 REM SUBR HASH FUNCTION H(X$)
1100 DEF FNH(X$)=LEN(X$)*5+ASC(LEFT$(X$,1))+ASC(RIGHT$(X$,1))*100
1110 E=300
1120 FOR K1=1 TO E
1130    K$(K1)=""
1140    NEXT K1
1150 FOR K1=1 TO E
1160    INPUT#-1,N,K$(N),R$(N),R1$(N)
1170    IF N=0 THEN 1190
1180    NEXT K1
1190 FOR D=1 TO 5
1200    READ D$(D)
1210    NEXT D
1220 DATA ".",",","?",";","!"
1230 P$="ｱﾅﾀ ﾉ ﾅﾏｴ ﾊ? ｿｳﾀﾞﾝ ｺﾞﾄ ﾊ?"
1240 FOR T=1 TO 10000
1250    PRINT "C:";P$
1260    PRINT "M:";
1270    INPUT A$
1280    IF LEFT$(A$,1)="#" THEN 1630
1290    REM GAKUSHU
1300    X$=P$
1310    GOSUB 1750
1320    L$=LEFT$(L$,4)
1330    H=FNH(L$)
1340    FOR N=0 TO 10
1350       K1=FNK(H,N)
1360       IF K$(K1)="" THEN 1390
1370       NEXT N
1380    K1=FNK(H,INT(RND(1)*10))
1390    K$(K1)=L$
1400    R$(K1)=A$
1410    X$=A$
1420    GOSUB 1750
1430    L1$=LEFT$(L$,4)
1440    R1$(K1)=L1$
1450    REM OUTOU
1460    H=FNH(R1$(K1))
1470    FOR N=0 TO 10
1480       K1=FNK(H,N)
1490       IF K$(K1)="" THEN 1550
1500       IF K$(K1)<>L1$ THEN 1540
1510       IF RND(1)<.5 THEN 1540
1520       P$=R$(K1)
1530       GOTO 1620
```

図12-5　しゃべる機械のプログラム ①

```
1540        NEXT N
1550     P$=L$+"? "
1560     IF RND(1)<.5 THEN 1620
1570     FOR K2=INT(RND(1)*E+1) TO E
1580        IF K$(K2)="" THEN 1610
1590        P$=P$+R$(K2)
1600        GOTO 1620
1610        NEXT K2
1620     NEXT T
1630 REM SAVE
1640 PRINT "END. TAPE 0 JUNBI SEYO."
1650 INPUT A$
1660 N=0
1670 FOR K1=1 TO E
1680     IF K$(K1)="" THEN 1710
1690     PRINT#-1,K1,K$(K1),R$(K1),R1$(K1)
1700 N=N+1:PRINTN;
1710     NEXT K1
1720     PRINT#-1,0,"E","N","D"
1730 STOP
1740 REM---------------------------
1750 REM SUBR FIND THE LONGEST WORD L$(X$)
1760 M9=0
1770 X$=X$+" "
1780 A=0
1790 FOR L=1 TO LEN(X$)
1800     S$=MID$(X$,L,1)
1810     FOR D=1 TO 5
1820        IF S$<>D$(D) THEN 1840
1830        S$=" "
1840        NEXT D
1850     IF A=1 THEN 1900
1860     IF S$=" " THEN 1950
1870     A=1
1880     S9=L
1890     GOTO 1950
1900     IF S$<>" " THEN 1950
1910     A=0
1920     IF M9>=L-S9 THEN 1950
1930     M9=L-S9
1940     L$=MID$(X$,S9,M9)
1950     NEXT L
1960 RETURN
1970 END
```

図12-5 しゃべる機械のプログラム ②

課　題

12—1　ここでは直前の言葉に対する応答だけを学習するようになっていますが，さらにもっと前の言葉との対応を付けて学習するようにしてください。(解答207ページ)

12—2　文の中のもっとも長い文字列を取り出してキーにしていますが，次に長い文字列も取り出してキーにするようにしてください。(解答207ページ)

12—3　（フロッピーディスクがある場合）学習したデータを，すぐディスクに格納するようにしてください。データを取り出すのもディスクからにします。こうすると非常に多量の文を蓄えておくことができます。プログラムは多彩な反応ができるようになります。(解答207ページ)

13 「課題」の解答

1　論理パズル1――仲良し仲間が海へ行く

1―1　図1-1のプログラムのFOR　K＝…とNEXT　Kのループの中を除き，**図13-1**を入れれば，この問題のプログラムができあがります。

Wに海へ行く人数が入り，Wが奇数のときに，

W＜＞INT（W／2）＊2

が成り立つことに注意してください。

結果は**図13-2**のようになります。つまりアヤコさん，イクコさん，オウイチ君，カズオ君が行くということになります。

```
110        W=A+I+U+E+O+K
120        IF W<>INT(W/2)*2 GOTO 220
130        IF A+I+E<=1 GOTO 220
140        IF U+O+K<=1 GOTO 220
150        IF A>I GOTO 220
160        IF U>E GOTO 220
170        IF O>K GOTO 220
180        IF I*E*K=1 GOTO 220
```

図13-1　課題1―1のプログラム（変えたところだけ）

```
ﾛﾝﾘﾊﾟｽﾞﾙｰｰｰｳﾐ ﾍ ｲｸ
ｱﾔｺ    ｲｸｺ    ｳｳﾁﾔﾝ ｴｲｺ ｵｳｲﾁ ｶｽﾞｵ
 ｲｸ     ｲｸ    ｲｶﾅｲ ｲｶﾅｲ  ｲｸ    ｲｸ
```

図13-2　課題1―1の出力結果

1―2　上と同じように図1-1に**図13-3**を入れます。
結果は**図13-4**のようになります。組合せは，このように2通りあるわけです。この両方に行くのが**エイコさん**で，つまり「必ず行く」

ということになります。同じように「必ず残る」のは**オウイチ君**です。これだけでの条件では，ほかの4人は行くか，行かないかはまだわかりません。

```
110         IF A+2*I+3*U+4*E+5*0+6*K<>10 GOTO 220
120         IF A>I GOTO 220
130         IF U>E GOTO 220
140         IF 0>K GOTO 220
```

図13-3 課題1—2のプログラム(変えたところだけ)

```
ロンリパズルーーーウミ ヘ イク
アヤコ   イクコ   ウウチヤン エイコ オウイチ カズオ
イカナイ イカナイ イカナイ   イク   イカナイ  イク
アヤコ   イクコ   ウウチヤン エイコ オウイチ カズオ
 イク     イク     イク     イク   イカナイ イカナイ
```

図13-4 課題1—2の出力結果

2 論理パズル2——本を並べたりお見合いをさせたり

2—1 (お見合い) プログラムは**図13-5**のようになります。結果は**図13-6**です。こったやり方をしたのは，

IF (X(4)−3.5)＊(X(5)−3.5)＜0 THEN……

のところで，X(4)，X(5) が片方だけが**3.5**以上になる場合，つまり双子の一方だけが双子と見合いするという場合を除いています。

```
0010 PRINT "COUPLE O TSUKURU"
0020 DIM X(5),R(5),J$(5),D$(5)
0030 FOR M=1 TO 5
0040     READ J$(M),D$(M)
0050     NEXT M
0060 DATA "HITOMI  ","KAZUO   "
0070 DATA "FUMI    ","JIROU   "
0080 DATA "MIMI    ","SABUROU "
0090 DATA "YOSHIMI ","YOSHIO  "
0100 DATA "ITSUMI  ","GOROU   "
0110 FOR N=0 TO 119
0120     R1=N
0130     FOR M=1 TO 5
0140         R(6-M)=R1-INT(R1/M)*M
0150         R1=INT(R1/M)
0160         X(M)=M
0170         NEXT M
```

図13-5 課題2—1のプログラム(1)

```
0180  FOR M=1 TO 5
0190     W=X(M)
0200     X(M)=X(M+R(M))
0210     X(M+R(M))=W
0220     NEXT M
0230   IF (X(4)-3.5)*(X(5)-3.5)<0 THEN 330
0240   IF X(4)>X(5) THEN 330
0250   IF X(1)>X(2) THEN 330
0260   IF X(2)>X(3) THEN 330
0270   IF (X(1)-2)*(X(2)-2)*(X(3)-2)=0 THEN 330
0280   IF X(4)=1 THEN 330
0290   PRINT "COUPLE: ONNA      OTOKO"
0300   FOR M=1 TO 5
0310     PRINT "        (";J$(M);",";D$(X(M));")"
0320     NEXT M
0330   NEXT N
0340 END
```

図13-5　課題２―１のプログラム(2)

```
COUPLE O TSUKURU
COUPLE: ONNA      OTOKO
        (HITOMI  ,KAZUO   )
        (FUMI    ,YOSHIO  )
        (MIMI    ,GOROU   )
        (YOSHIMI ,JIROU   )
        (ITSUMI  ,SABUROU )
```

図13-6　課題２―１の出力結果

２―２　(一郎二郎……)　ここでは出てくる人数（あるいは本の数等）を自由に変えられるようにしました。プログラムは**図13-7**です。

IF　ABS(X(M)−X(M−１))＜＝**2**　THEN～

の行で初めに隣り合っていた２人が距離２以下になる場合を除いています。X(M) はM郎の場所です。表示のときは並んだ順を出すようにしてあります。答は２通りですが，左右の対称性を考えれば１通りで，５３１６４２の順ということになります。

２―３　(数を並べる)　プログラムは課題２―２とほとんど同じですが，条件のところだけ変えればよいのです。答は対称性を除けば，

４１５２６３,

５３１６４２

の２通りになります。

```
0010 PRINT "ICHIROU NO KYOUDAI: 2-2"
0020 DIM X(10),R(10)
0030 M1=6
0040 N1=1
0050 FOR N3=1 TO M1
0060    N1=N1*N3
0070    NEXT N3
0080 FOR N=0 TO N1
0090    R1=N
0100    FOR M=1 TO M1
0110       R(M1+1-M)=R1-INT(R1/M)*M
0120       R1=INT(R1/M)
0130       X(M)=M
0140       NEXT M
0150    FOR M=1 TO M1
0160       W=X(M)
0170       X(M)=X(M+R(M))
0180       X(M+R(M))=W
0190       NEXT M
0200    FOR M=2 TO M1
0210       IF ABS(X(M)-X(M-1))<=2 THEN 300
0220       NEXT M
0230    FOR M=1 TO M1
0240       H(X(M))=M
0250       NEXT M
0260    PRINT
0270    FOR M=1 TO M1
0280       PRINT H(M);"ROU ";
0290       NEXT M
0300    NEXT N
0310 END
```

図13-7 課題２—２のプログラム

```
ICHIROU NO KYOUDAI: 2-2

    5 ROU     3 ROU     1 ROU     6 ROU     4 ROU     2 ROU
    2 ROU     4 ROU     6 ROU     1 ROU     3 ROU     5 ROU
```

図13-8 課題２—２の出力結果

先日，筆者の参加しているパズル懇和会でこの問題のことを話したら，「8-クイーン」の問題と同じことになるということを教えて下さった方がいました。よく考えてみるとそうなっています。8-クイーンは盤の上にクイーン（8方向に進める駒）を互いに取られないように置くという問題です。

どうして同じなのか考えてみてください。

```
0010 PRINT "KAZU O NARABERU"
0020 DIM X(10),R(10)
0030 FOR M1=1 TO 7
0040    PRINT M1;"KO NO KAZU:"
0050    N1=1
0060    FOR N3=1 TO M1
0070       N1=N1*N3
0080       NEXT N3
0090    FOR N=0 TO N1
0100       R1=N
0110       FOR M=1 TO M1
0120          R(M1+1-M)=R1-INT(R1/M)*M
0130          R1=INT(R1/M)
0140          X(M)=M
0150          NEXT M
0160       FOR M=1 TO M1
0170          W=X(M)
0180          X(M)=X(M+R(M))
0190          X(M+R(M))=W
0200          NEXT M
0210       FOR M2=1 TO M1
0220          FOR M3=M2+1 TO M1
0230             IF ABS(X(M2)-X(M3))=M3-M2 THEN 330
0240             NEXT M3
0250          NEXT M2
0260       FOR M=1 TO M1
0270          H(X(M))=M
0280          NEXT M
0290       FOR M=1 TO M1
0300          PRINT H(M);
0310          NEXT M
0320       PRINT
0330       NEXT N
0340    NEXT M1
0350 END
```

図13-9　課題２—３，２—４のプログラム

2—4　課題２—３の人数を変えられるようにし，１人から７人までを調べたのが**図13-9**，**図13-10**です。人数が７人になるとたくさんの解があります。課題２—２も同じ方法で行なえばよいのです。

```
KAZU O NARABERU
   1 KO NO KAZU:
   2 KO NO KAZU:
   3 KO NO KAZU:
   4 KO NO KAZU:
   3     1     4     2
   2     4     1     3
   5 KO NO KAZU:
   1     4     2     5     3
   1     3     5     2     4
   3     1     4     2     5
   4     1     3     5     2
   2     4     1     3     5
   2     5     3     1     4
   4     2     5     3     1
   3     5     2     4     1
   6 KO NO KAZU:
   4     1     5     2     6     3
   5     3     1     6     4     2
   2     4     6     1     3     5
   3     6     2     5     1     4
   7 KO NO KAZU:
   1     5     2     6     3     7     4
   1     6     4     2     7     5     3
   1     3     5     7     2     4     6
   1     4     7     3     6     2     5
   3     1     6     2     5     7     4
   4     1     5     2     6     3     7
   4     1     3     6     2     7     5
   3     1     6     4     2     7     5
   5     1     6     4     2     7     3
```

図13-10 課題２—３，２—４の出力結果

２—５　(環状にする) 両端をつなぎ合わせるようにするわけです。課題２—３で行なうと**図13-11**のようになります。この結果は**図13-12**のようになります。

```
0010 PRINT "ICHIROU NO KYOUDAI: 2-2"
0020 DIM X(10),R(10)
0030 FOR M1=1 TO 8
0040    PRINT "NINZUU";M1
0050    N1=1
0060    FOR N3=1 TO M1
0070       N1=N1*N3
0080       NEXT N3
0090    FOR N=0 TO N1
0100       R1=N
0110       FOR M=1 TO M1
0120          R(M1+1-M)=R1-INT(R1/M)*M
0130          R1=INT(R1/M)
0140          X(M)=M
0150          NEXT M
0160       FOR M=1 TO M1
0170          W=X(M)
0180          X(M)=X(M+R(M))
0190          X(M+R(M))=W
0200          NEXT M
0210       FOR M=2 TO M1
0220          IF ABS(X(M)-X(M-1))<=2 THEN 340
0230          IF ABS(X(M)-X(M-1))>=M1-2 THEN 340
0240          NEXT M
0250       IF ABS(X(1)-X(M1))<=2 THEN 340
0260       IF ABS(X(1)-X(M1))>=M1-2 THEN 340
0270       FOR M=1 TO M1
0280          H(X(M))=M
0290          NEXT M
0300       FOR M=1 TO M1
0310          PRINT H(M);"ROU ";
0320          NEXT M
0330       PRINT
0340       NEXT N
0350    NEXT M1
0360 END
```

図13-11 兄弟を並ばせる問題を環状にしたもの

2―6 （国連討議）中身は「男女の組を作る」の例題と同じです。わかりましたか？

2―7 たとえば1が中央より左にある場合（1が中央なら2が中央より左にある場合）だけを選べばいいのです。

2―8 1つの数字，たとえば1を最初の位置に固定してしまえばいいのです。

```
ICHIROU NO KYOUDAI: 2-2
NINZUU    1
NINZUU    2
NINZUU    3
NINZUU    4
NINZUU    5
NINZUU    6
NINZUU    7
   1 ROU     6 ROU     4 ROU     2 ROU     7 ROU     5 ROU     3 ROU
   1 ROU     3 ROU     5 ROU     7 ROU     2 ROU     4 ROU     6 ROU
   3 ROU     1 ROU     6 ROU     4 ROU     2 ROU     7 ROU     5 ROU
   6 ROU     1 ROU     3 ROU     5 ROU     7 ROU     2 ROU     4 ROU
   5 ROU     3 ROU     1 ROU     6 ROU     4 ROU     2 ROU     7 ROU
   4 ROU     6 ROU     1 ROU     3 ROU     5 ROU     7 ROU     2 ROU
   2 ROU     4 ROU     6 ROU     1 ROU     3 ROU     5 ROU     7 ROU
   7 ROU     5 ROU     3 ROU     1 ROU     6 ROU     4 ROU     2 ROU
   7 ROU     2 ROU     4 ROU     6 ROU     1 ROU     3 ROU     5 ROU
   2 ROU     7 ROU     5 ROU     3 ROU     1 ROU     6 ROU     4 ROU
   4 ROU     2 ROU     7 ROU     5 ROU     3 ROU     1 ROU     6 ROU
   5 ROU     7 ROU     2 ROU     4 ROU     6 ROU     1 ROU     3 ROU
   6 ROU     4 ROU     2 ROU     7 ROU     5 ROU     3 ROU     1 ROU
   3 ROU     5 ROU     7 ROU     2 ROU     4 ROU     6 ROU     1 ROU
NINZUU    8
   1 ROU     4 ROU     7 ROU     2 ROU     5 ROU     8 ROU     3 ROU
   6 ROU
   1 ROU     4 ROU     6 ROU     2 ROU     8 ROU     5 ROU     3 ROU
   7 ROU
```

図13-12　図13-1のプログラムの結果

3　虫食算を解く

3—1　答は**図13-13**の通りです。(次ページ)

3—2　プログラムを**図13-14**に示します。実は課題3—1はこのプログラムで作ったものです。

方法としては次のようにしています。まず全部が虫食の状態から始めます。1つ数字を入れた問題を考えます。解いてみて解が2個以上あったらさらに数字を入れます。解が1個になったらでき上がりです。解がなくなったら数字を1つ再び虫食にもどします。

```
=================
 NO.   1

          9 9
          3 9
-----------------
        8 9 1
      2 9 7
-----------------
      3 8 6 1
=================
 NO.   2

          2 9
          4 7
-----------------
        2 0 3
      1 1 6
-----------------
      1 3 6 3
=================
 NO.   3

          6 2
          4 2
-----------------
        1 2 4
      2 4 8
-----------------
      2 6 0 4
=================
 NO.   4

          9 7
          9 7
-----------------
        6 7 9
      8 7 3
-----------------
      9 4 0 9
=================
 NO.   5

          8 8
          4 2
-----------------
        1 7 6
      3 5 2
-----------------
      3 6 9 6
=================
 NO.   6

          9 3
          5 4
-----------------
        3 7 2
      4 6 5
-----------------
      5 0 2 2
=================
 NO.   7

          3 9
          3 7
-----------------
        2 7 3
      1 1 7
-----------------
      1 4 4 3
```

```
=================
 NO.   8

          5 3
          5 3
-----------------
        1 5 9
      2 6 5
-----------------
      2 8 0 9
=================
 NO.   9

          3 3
          6 7
-----------------
        2 3 1
      1 9 8
-----------------
      2 2 1 1
=================
 NO.  10

          5 4
          2 9
-----------------
        4 8 6
      1 0 8
-----------------
      1 5 6 6
=================
 NO.  11

          5 7
          7 7
-----------------
        3 9 9
      3 9 9
-----------------
      4 3 8 9
=================
 NO.  12

          9 8
          7 5
-----------------
        4 9 0
      6 8 6
-----------------
      7 3 5 0
=================
 NO.  13

          5 7
          5 7
-----------------
        3 9 9
      2 8 5
-----------------
      3 2 4 9
=================
 NO.  14

          3 0
          8 4
-----------------
        1 2 0
      2 4 0
-----------------
      2 5 2 0
```

```
=================
 NO.  15

          9 0
          8 2
-----------------
        1 8 0
      7 2 0
-----------------
      7 3 8 0
=================
 NO.  16

          8 4
          8 5
-----------------
        4 2 0
      6 7 2
-----------------
      7 1 4 0
=================
 NO.  17

          7 2
          2 8
-----------------
        5 7 6
      1 4 4
-----------------
      2 0 1 6
=================
 NO.  18

          6 9
          6 9
-----------------
        6 2 1
      4 1 4
-----------------
      4 7 6 1
=================
 NO.  19

          9 3
          7 3
-----------------
        2 7 9
      6 5 1
-----------------
      6 7 8 9
=================
 NO.  20

          9 6
          9 6
-----------------
        5 7 6
      8 6 4
-----------------
      9 2 1 6
=================
 NO.  21

          8 5
          3 9
-----------------
        7 6 5
      2 5 5
-----------------
      3 3 1 5
```

```
=================
 NO.  22

          9 0
          8 5
-----------------
        4 5 0
      7 2 0
-----------------
      7 6 5 0
=================
 NO.  23

          4 9
          4 7
-----------------
        3 4 3
      1 9 6
-----------------
      2 3 0 3
=================
 NO.  24

          6 2
          6 9
-----------------
        5 5 8
      3 7 2
-----------------
      4 2 7 8
```

図13-13　虫食算の答

```
0010 REMPRINT "MUSHIKUIZAN"
0020 DIM X(10),Y(10),Z(10,11),U(20)
0030 DIM L$(10)
0040 REM 10^(K-1) NO KETA
0050 DEF FNA(W1)=W1-INT(W1/10)*10
0060 DEF FNK(W,K)=FNA(INT(W/10^(K-1)))
0070 FOR K=0 TO 10
0080    READ L$(K)
0090    NEXT K
0100 DATA " []","  0","  1","  2","  3","  4"
0110 DATA       "  5","  6","  7","  8","  9"
0120 NO=0
0130 M=2
0140 N=2
0150 R=M+N+(M+1)*N+M+N
0160 FOR A1=1 TO 50
0170    REMPRINT "TRAL NO.";A1
0180    FOR M1=1 TO M
0190       X(M1)=-1
0200       NEXT M1
0210    FOR N1=1 TO N
0220       Y(N1)=-1
0230       FOR M1=1 TO M+1
0240          Z(N1,M1)=-1
0250          NEXT M1
0260       NEXT N1
0270    FOR L=1 TO M+N
0280       U(L)=-1
0290       NEXT L
0300    T=2
0310    REM START
0320    FOR A=1 TO R
0330       REMPRINT "<";A;" カイメ>  #";T;".  ";
0340       IF T=1 THEN 430
0350       IF T=0 THEN 400
0360       R1=INT(RND(-1)*R)+1
0370       C=INT(RND(-1)*10)
0380       GOSUB 1400
0390       GOTO 450
0400       C=-1
0410       GOSUB 1400
0420       GOTO 450
0430       GOSUB 840
0440       GOTO 820
0450       T=0
0460       REM KEISAN
0470       FOR X9=10^(M-1) TO 10^M-1
0480          FOR M1=1 TO M
0490             IF X(M1)=-1 GOTO 510
0500             IF FNK(X9,M1)<>X(M1) GOTO 790
0510             NEXT M1
0520          FOR Y9=10^(N-1) TO 10^N-1
0530             FOR N1=1 TO N
0540                IF Y(N1)=-1 GOTO 560
```

図13-14 虫食算を作るプログラム(1)

```
0550               IF FNK(Y9,N1)<>Y(N1) GOTO 780
0560               Z9=FNK(Y9,N1)*X9
0570               FOR M1=1 TO M+1
0580                  IF Z(N1,M1)=-1 GOTO 600
0590                  IF FNK(Z9,M1)<>Z(N1,M1) GOTO 780
0600                  NEXT M1
0610               IF Z(N1,M+1)<>-1 THEN 630
0620               IF FNK(Z9,M+1)=0 THEN 780
0630               NEXT N1
0640            U9=X9*Y9
0650            FOR L=1 TO M+N
0660               IF U(L)=-1 GOTO 680
0670               IF FNK(U9,L)<>U(L) GOTO 780
0680               NEXT L
0690            IF U(M+N)<>-1 THEN 710
0700            IF FNK(U9,M+N)=0 THEN 780
0710            REM DEKIAGARI
0720            T=T+1
0730            IF T>=2 THEN 800
0740            REMPRINT X9;" *";Y9;" =";U9;
0750            X8=X9
0760            Y8=Y9
0770            U8=U9
0780            NEXT Y9
0790         NEXT X9
0800      REMPRINT
0810      NEXT A
0820   NEXT A1
0830 STOP
0840 REMSUBR PRINT
0850 PRINT "============================================"
0860 NO=NO+1
0870 PRINT "NO.";NO;
0880 J=M+N+2
0890 GOSUB 1570
0900 PRINT "NO.";NO
0910 PRINT
0920 J=N
0930 GOSUB 1570
0940 FOR M1=1 TO M
0950    PRINT L$(1+X(M+1-M1));
0960    NEXT M1
0970 J=N+4
0980 GOSUB 1570
0990 FOR M1=1 TO M
1000    PRINT L$(1+FNK(X8,M+1-M1));
1010    NEXT M1
1020 PRINT
1030 J=M
1040 GOSUB 1570
1050 FOR N1=1 TO N
1060    PRINT L$(1+Y(N+1-N1));
1070    NEXT N1
1080 J=M+4
```

図13-14 虫食算を作るプログラム(2)

```
1090 GOSUB 1570
1100 FOR N1=1 TO N
1110    PRINT L$(1+FNK(Y8,N+1-N1));
1120    NEXT N1
1130 PRINT
1140 PRINT "--X-----------------------------------"
1150 FOR N1=1 TO N
1160    J=N-N1
1170    GOSUB 1570
1180    FOR M1=1 TO M+1
1190       PRINT L$(1+Z(N1,M+2-M1));
1200       NEXT M1
1210    J=N-1+4
1220    GOSUB 1570
1230    FOR M1=1 TO M+1
1240       PRINT L$(1+FNK(FNK(Y8,N1)*X8,M+2-M1));
1250       NEXT M1
1260    PRINT
1270    NEXT N1
1280 PRINT "--------------------------------------"
1290 PRINT "   ";
1300 FOR L=1 TO M+N
1310    PRINT L$(1+U(M+N+1-L));
1320    NEXT L
1330 J=0+4
1340 GOSUB 1570
1350 FOR L=1 TO M+N
1360    PRINT L$(1+FNK(U8,M+N+1-L));
1370    NEXT L
1380 PRINT
1390 RETURN
1400 REM RANDOM SET
1410 REMPRINT " ﾊﾞｼﾖ";R1;" ﾆ";C;
1420 R2=R1
1430 IF R2>M GOTO 1460
1440 X(R2)=C
1450 RETURN
1460 R2=R2-M
1470 IF R2>N GOTO 1500
1480 Y(R2)=C
1490 RETURN
1500 R2=R2-N
1510 IF R2>(M+1)*N GOTO 1540
1520 Z(INT((R2-1)/N)+1,R2-INT((R2-1)/N)*N)=C
1530 RETURN
1540 R2=R2-(M+1)*N
1550 U(R2)=C
1560 RETURN
1570 REMSUBR PR.SPACE
1580 FOR J1=1 TO J+1
1590    PRINT "   ";
1600    NEXT J1
1610 RETURN
1620 END
```

図13-14 虫食算を作るプログラム(3)

3—3　一例としては次のようにできます。

```
FOR  U=0  TO  1
  FOR  O=0  TO  1
    IF  U<O  THEN
    FOR  A=0  TO  1
      IF  A+O=0  THEN
      FOR  K=0  TO  1
        IF  A=K  THEN
          ⋮
        NEXT  K
      NEXT  A
    NEXT  O
  NEXT  U
```

4　覆面算を解く

4—1　たし算の覆面算は，かけ算の場合よりもやさしいと思います。またサブルーチンも共用できます。

4—2　この問題では，おのおの約$10000^2 \fallingdotseq 10^8$通りおよび$10000^2 = 10^{10}$通り調べなければならないことになり，そのままの方法では前に説明した，一生かかってもできない話になってしまいます。

この問題には虫食はありませんから，覆面の記号に対応させる数字を変えることで，全部の場合を調べるのがよいと思います。最大10種類の覆面に対して0から9までの数字を重複しないで割り当てるわけです。結局，本棚（お見合い）のパズルと同じ形になります。この場合，組合せの数は，10！ですから，なんとか計算できる範囲内といえます。

4—3　次のようにすればよいでしょう。数字のあてはめのサブルーチン中で，

```
IF  Q=99  THEN
   ⋮
C=1
RETURN
```

の部分を，

```
IF  Q=99  THEN
   ⋮
IF  K1<>3  THEN
C=0
RETURN
C=1
RETURN
```

とする。

4－4　課題4－3の　IF　K1<>3　THEN～のところを，

```
IF  K1<>INT(K1/2)*2  THEN～
```

とすればよい。

5　切符の数字――数楽パズル

5－1　目的となる数Yを入力する代わりにループ，

```
FOR  Y=0  TO  100
  ⋮  }処理
  NEXT  Y
```

を使えばよいのです。また，答を1通りしか出さないために，

```
  ⋮
 [答の表示]
  GOTO
  ⋮
  NEXT  S1
PRINT  Y;"  コタエ  ナシ。"
NEXT  Y
```

のようにすればよいでしょう。結果を**図13-15**（次ページ）に示します。

5―2　虫食算，覆面算で使った，数の途中の桁を取り出す関数FNKを使ってみます。これでX(1)，…，X(4)を決めます。すなわち，

```
DEF  FNA(W1)=W1-INT(W1/10)*10
DEF  FNK(W,K)=FNA(INT(W/10^(K-1)))
  ⋮
Y=10
FOR  X1=1980  TO  2001
  FOR  K1=1  TO  4
    X(K1)=FNK(X1, 5-K1)
    NEXT  K1
    ⋮    } 処理

  NEXT  X1
```

とすればできます。

5―3　課題5―2とほとんど同じです。

```
FOR  X1=1980  TO  2001
```

の代わりに，

```
FOR  X1=1926  TO  2025
   Y=X1-1925
```

とします。

5―4　＋，－，＊，／の他に，

$12 \otimes 34 = 1234$

となるような計算⊗を付け加え，＋や－など同じように扱えればよいのです。ただし⊗の計算を後で行なってはいけない，つまり，

$(1+2) \otimes (3 \times 4)$

のようには使えないことに注意してください。

```
KIPPU PUZZLE: SUURAKU
4 KO NO KAZU?1,9,8,4
   0  = -(((    1   -      9 ) +      8 ) *      4  )
   1    コタエ ナシ.
   2  = -(      1   + (    9   - (    8   +      4 )))
   3  = -(      1   -((    9   -      8 ) *      4 ) )
   4  =   ((    1   -      9 ) +      8 ) +      4
   5  =         1   +((    9   -      8 ) *      4 )
   6  =   ((    1   +      9 ) -      8 ) +      4
   7  =         1   * (    9   - (    8   /      4 ))
   8  =   ((    1   +      9 ) -      8 ) *      4
   9    コタエ ナシ.
  10  = -(      1   - (    9   + (    8   /      4 )))
  11  =         1   * (    9   + (    8   /      4 ))
  12  =         1   + (    9   + (    8   /      4 ))
  13  =   ((    1   *      9 ) +      8 ) -      4
  14  =   ((    1   +      9 ) +      8 ) -      4
  15    コタエ ナシ.
  16  = -(((    1   -      9 ) *      8 ) /      4  )
  17  = -(      1   - (    9   * (    8   /      4 )))
  18  =   ((    1   *      9 ) *      8 ) /      4
  19  =         1   + (    9   * (    8   /      4 ))
  20  =   ((    1   +      9 ) *      8 ) /      4
  21  =   ((    1   *      9 ) +      8 ) +      4
  22  =   ((    1   +      9 ) +      8 ) +      4
  23  = -(      1   * (    9   - (    8   *      4 )))
  24  =    (    1   -      9 ) + (    8   *      4 )
  25    コタエ ナシ.
  26    コタエ ナシ.
  27    コタエ ナシ.
  28    コタエ ナシ.
  29    コタエ ナシ.
  30    コタエ ナシ.
  31    コタエ ナシ.
  32  = -( (     1   -      9 ) * (    8   -      4 ) )
  33    コタエ ナシ.
  34    コタエ ナシ.
  35  = -(       1   - (    9   * (    8   -      4 )))
  36  =          1   * (    9   * (    8   -      4 ))
  37  =          1   + (    9   * (    8   -      4 ))
  38    コタエ ナシ.
  39    コタエ ナシ.
  40  =    (     1   +      9 ) * (    8   -      4 )
  41  =          1   * (    9   + (    8   *      4 ))
  42  =          1   + (    9   + (    8   *      4 ))
  43    コタエ ナシ.
  44    コタエ ナシ.
  45    コタエ ナシ.
  46    コタエ ナシ.
  47    コタエ ナシ.
  48    コタエ ナシ.
  49    コタエ ナシ.
```

⋮

図13-15　0から100まで作る

5—5　4個の数字をX(1)，……，X(4)に割り当ててから調べればよいわけです。4！＝24通りの割り当て方があります。これは本棚(お見合い) の問題と同じです。

6　幸福の手紙——自分を表示するプログラム

6—1　きっと次のようになるでしょう。1回目は行番号30の行と130の行が，

```
 30  DATA  PRINT  N;"DATA";"PRINT  123"
130  PRINT  N;"DATA";"PRINT  123"
```

となります。2回目は10～100の行が，

```
 10  DATA  PRINT  123
          ⋮
100  DATA  PRINT  123
```

となってしまいます。3回目は110～200の行が，

```
110  PRINT  123
 ⋮
200  PRINT  123
```

となります。最後には‘123’を10回表示します。これ以上はいきません。だんだんくずれていくところに奇怪さを感じますね。

6—2　行番号を1，2，3，…とすると，STEPのところがいらなくなります。1行に多くの文を書くことを許すことも考えられます。

6—3　次のようなものはどうでしょうか。

```
 10  DATA   ┌───┐
  ⋮         │ ※ │
120  DATA   └───┘
```

※下の130から240までが入る

```
130 | DIM  A$(12)
140 | FOR  N=1  TO  12
150 | READ  A$(N)
```

```
160 NEXT N
170 FOR N=1 TO 12
180 PRINT (25-N)*10;A$(13-N)
190 NEXT N
200 FOR N=1 TO 12
210 PRINT (13-N)*10;" DATA ";A$(13-N)
220 NEXT N
230 REM GYAKU HYOUJI
240 END
```

7 桂馬道の問題

7—1 盤の上で行けるところを0，行けないところを−1で表わすことにします。図に対応して配列B1にこのデータを入力します。

```
N3=0
FOR X=1 TO X3
  FOR Y=1 TO Y3
    READ B1(X, Y)
    N3=B1(X, Y)+1+N3
    NEXT Y
  NEXT X
DATA -1, -1, -1, 0, -1, -1, -1
DATA -1, -1,  0, 0,  0, -1, -1
DATA -1,  0,  0, 0,  0,  0, -1
                ⋮
DATA -1, -1, -1, 0, -1, -1, -1
```

さらにBの初期設定はB(X, Y)＝0の代わりに，

B(X, Y)＝B1(X, Y)

とします。

上のN3には終了までの手数が入ることになりますから，終了条件のところのX3＊Y3をN3にすればうまく動きます。

7－2　問題7－1のREAD　B1(X, Y) のところを,

　　B1(X, Y)＝INT(RND(1)＋P)－1

とすれば，B1(X, Y)は確率Pで行ける部分(0)，確率1－Pで行けない部分(－1)になります。RND関数は0から1までの一様乱数を与えます。Pは0.8から0.9ぐらいにするとよいでしょう。

7－3　変数D，配列X8，Y8への入力データ文を変えれば実現できます。これらが駒の動き方を表わすからです。銀ならば,

```
DATA  5
DATA  -1, -1, -1, 0, -1, 1, 1, -1, 1, 1
```

とすればよいのです。駒の動きをデータで表わすことにしてあると，このように簡単にできることになります。

7－4　終了条件の判定のところで，現在の位置(X, Y)から最初の位置(X4, Y4)に行けるかどうか，ということを付け加えます。

終了条件を,

```
REM  OWARIKA
F=0
IF  N<X3*Y3  THEN ──────────────┐
FOR  D1=1  TO  D                 │
    IF  X+X8(D1)<>X4  THEN ──┐   │
    IF  Y+Y8(D1)<>Y4  THEN ──┤   │
    F=1                      │   │
    RETURN                   │   │
    NEXT  D ◄────────────────┘   │
RETURN ◄─────────────────────────┘
```

とすればよいでしょう。速くする工夫の余地はあります。

7－5　おのおのの位置に対して何通り進む方向があるか，ということを与える配列Sを用意します。進みながらこの値を書き換えていきます。袋小路はS(X, Y) の値が1，孤立点は0となります。

ある場所へ進んだとき，

(1)　S(X,Y)の値が1となる位置が1個できたならば，次にそちらへ進む。

(2)　S(X,Y)が1となるような位置が2個以上できた場合は，失敗なのでもどる。

ようにします。

8　碁石パズル——「ひろいもの」

8—1　直前の進む方向はD9(N)で，次の進む方向はD1です。直進してはいけないということから，

D9(N)≠D1

です。また，もどれないということも考えに入れると，ひろえるかどうか調べるサブルーチンの中の，

```
IF  ABS(D9(N)-D1)=2  THEN~
```

のところを，

```
IF  D9(N)-D1=INT((D9(N)-D1)/2)*2  THEN~
```

に取り換えればよいでしょう。

8—2　DとX8，Y8に読み込ませるデータを，

```
DATA  8
DATA  0,  1,  1,  1,  1, 0,  1, -1
DATA  0, -1, -1, -1, -1, 0, -1,  1
```

こうすれば8方向に行けるようになります。後もどりできないようにしておくところ，

```
IF  ABS(D9(N)-D1)=2  THEN~
```

の2のところを4にしておきます。

8—3　今まで石のあったところを通ってはいけない場合ですが，その場所には何番目に通ったかという，正の数が入っているということを使えばよいでしょう。つまりひろえるかどうかのサブルーチン

の中で，

IF　B(X１，Y１)＜＞０　THEN～

の行の前に，

IF　B(X１，Y１)＞０　THEN（RETURNの行番号)

とすればできます。

８—４　課題８—３とは逆にB(X１，Y１)が－９の場合，通れません。つまり上と同じところに，

IF　B(X１，Y１)＜０　THEN（RETURNの行番号)

を入れればよいのです。

9　迷路パズル——囲碁の表示システムを作る

９—１　ヘックスの盤は正三角形がつながったものです。これを二次元の配列であつかうときは，**図13-16**のように少しずらして考えます。つながり方は配列X９，Y９で表わせます(**図13-17**)。前の課題にあるナイトの動きを銀の動きに変える話と同じように，X９，Y９の中身を変えればいいのです（おのおの長さ６にします)。

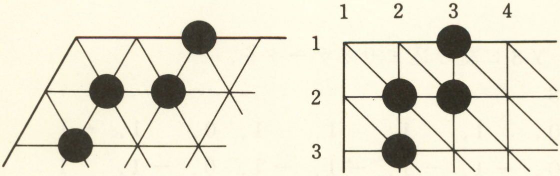


図13-16　ヘックス盤を二次元配列で表わす

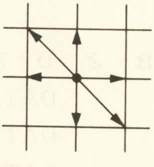


図13-16　ヘックスのつながり方

９—２　「進む方向」を状態の一部として覚えておかなければなりません。つまり **X座標，Y座標，方向**の３つで現在の状態が決まります。これらをX，Y，D１で表わすと，D方向へ進む状態の変化は，

D１＝D

X＝X＋X９(D)

Y＝Y＋Y 9 (D)

で表わせます。

右に曲がるということは，方向D 1から見て方向Dが逆時計回りに90°の角度があるということで表わせます。その場合は進めないことにします。

9－3 ぬりかえのサブルーチンのところで，1つの変数を用意しておき，1個の場所をぬりかえるたびに1を加えるようにします。この値の分だけ，あとで取り石の数に加えます。

9－4 劫かどうかの判定法は，次のようにできます。**図13-18**を見てください。

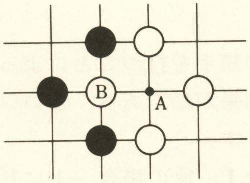


図13-18 劫（こう）

Aに黒が打つ手が劫の禁手であるということは，

(1) 直前に白がAのとなり(B)に打った。

(2) その手でAにあった黒石が取られた。

(3) A点はBの位置以外は直接白に囲まれている。

(4) B点はAの位置以外は直接黒に囲まれている。

の全部が満たされることです。(1)，(2)はスタックのトップを見ればわかります。(3)，(4)は盤面を見ます。

9－5 置碁もできるようにすること。画面のカーソルで打つ場所を入力すること。いろいろあると思います。

また，記憶させておいた手を順に出すようにして，囲碁練習機にすることも考えられます。ここから先は容易ですから，ぜひやって

みてください。

10　詰碁を解く

10—1　劫のことを何もしていないと，場合によっては，劫を取ったり取られたりが無限（といってもスタックがオーバーフローするまで）に続きます。

セキの場合は問題ありません。両方が手を出せない，つまりパスすることになるので，**生き**です。

10—2　前の章の課題で劫について説明したことを，やはりしなくてはいけません。ただし，劫を単に禁手にするのではなく，

(1)　黒（先手）が劫の禁手を打つことによって，黒の成功（黒生か白死）になる。

(2)　白（後手）が劫の禁手を打つことによって，黒の失敗となる。

とすれば，その詰碁の結果が劫である，といいます。これを普通，「黒先劫」の問題といいます。

評価（V）を勝の場合1，負の場合 −1 にしてきましたが，劫ならばこの中間の「0」といった値にするのがよいでしょう。

10—3　図10-4のプログラムでは，

```
0680　GOTO　710
```

の行によって，解が1個見つかれば終了し，次の問題に移るようになっています。これを取り去ればよいのです。

10—4　調べるべき手（X7(1)，Y7(1))，…，(X7(N7)，Y7(N7))のおのおのについて，その読むべきかどうかの価値を計算し，その高いものから先に調べる，というのが1つの方法です。評価の基準としては，次のようなものが考えられるでしょう。

(1)　(1，2)の点に高い評価を与える（格言にもある)。

(2)　まわりに石がないところに高い評価を与える。

(3)　石を切る点に高い評価を与える。

このような方法を実施するにあたっては，どの手をすでに読んだか，ということをスタックに保存しておく必要があることに注意してください。

11 麻雀のパズル

11—1 細かい点を除けば，本文通りできるはずです。

11—2 やってみてください。その他に，

(V) 孤立牌が2個以上あればテンパイしていない。

などもあるでしょう。

孤また麻雀をプレーするプログラムを作る場合には，直前の情報を使ってテンパイ探索を効率化できます。たとえば，直前が自模(ツモ)切りならば，直前の情報をそのまま活用できます。

11—3 一シャンテンとは，1個の牌を交換したのち，テンパイにになっている状態です。つまり2個の牌をもってきて，1個を捨てた形が上がり形になっていればよいのです。テンパイ探索のプログラムの中で「1個つけ加える」の部分を「2個つけ加えて1個捨てる」とします。

ここで計算量が非常にふえるということに注意してください。清一色なら「1個つけ加える」というのが9通りでしたが,「2個つけ加えて1個捨てる」というのは，

$_9C_2 \cdot 7 + 9 \cdot 8 = 324$通り

と36倍になります。一般の場合も，そのままの方法なら，約500倍の場合の数があります。課題11—2で説明した効率化の手法をやってみる必要があるでしょう。

11—4 図11-4のプログラムでは，L1が面子の位置，配列Sが面子が対子か順子かを表わしています。これを使って役を調べることができます。たとえば，

(1) 一ぺーコー：

$L1(i)=L1(j)$ かつ $S(i)=S(j)=1$

（ただし $i\neq j$）となる i，j がある。

(2) 一気通貫：

$L1(i)=1$，$L1(j)=4$，$L1(k)=7$，

$S(i)=S(j)=S(k)=1$

となる i，j，k がある。

(3) 四暗刻：

$i=1$，2，3，4 に対し $S(i)=0$

(4) 平和：

$i=1$，2，3，4 に対し $S(i)=1$

かつ，ある j に対し上がり牌A1が，

$L1(j)=A1\neq 7$ または $L1(j)+2=A1\neq 3$

になる（両面待ち）。

このように配列L1とSを使えば一般の場合でも，すっきり表わせます。

点の計算にはいろいろ面倒な部分がありますが，だいたいのところは，役の数をY，符の数をFとすると，

INT((INT((F＋9)/10)＊10＊2＾(Y＋2)＊X＋99)/100)＊100

で計算できます。ただしXは，

1　子と子の間の自模の受け渡しのとき
2　親と子の間の自模の受け渡しのとき
4　子への振り込み
6　親への振り込み

という値をとります。

11—5　34の長さの配列を用意します。また使いやすさから考えて，名前は2文字にすべきです。

11—6　配列Kの値が全部0か2なら七対子が完成しています。

11—7　上がり牌をA1とすると，

4－K(A 1)

が残っている牌の数です。全部の上がり牌に対するこの値をたせばよいのです。

11—8 牌の取り出し方に注意してください。つまり，3牌あるところに1牌もってくる確率と，ひとつもないところに1牌もってくる確率は違います。

すべての牌に番号を付けておきます。同じ種類のものにも，おのおの違う番号を付けます。この番号に従って重複しないように13個取り出せばよいのです。

12 しゃべる機械

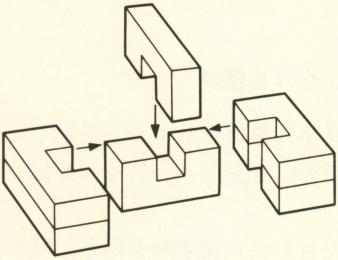
12—1 前に調べた最も長い文字列を保存しておく。新しく作った最も長い文字列とともに，相手の応答を組み合わせ，記憶します。

12—2 文の中から取り出した最も長い文字例を除き，残りの部分からもう一度最も長い文字列を取り出せばよいでしょう。

12—3 普通ランダムファイルを使います。そして，毎回ディスクを読んだり，ディスクに書いたりするのが簡単です。こうすると応答も速く，終わったときに，記憶したものをしまうための時間をかける必要がありません。

14 「休憩室」の解答

休憩室 1　休むどころか疲れる問題ですね。ペンタキューブのようなブロックパズルを解くプログラムもよく作られています。



コ型ペンタキューブ
6個による直方体

休憩室 3　結果は多分，パソコンの機種によっても違いますが，

1　2　3　4　……　20
21　22　……
41　42　……
⋮
××××
ERROR：STACK　OVERFLOW

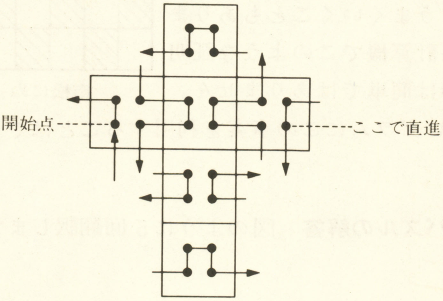
というようなものが出てくると思います。

なぜこうなるかというと，GOSUBはGOTOと違ってサブルーチンのもどり先をスタックにとっておくのです。くわしくいうと，GOSUB αと書かれたところに来ると，その文の次の文の場所の情報をスタックにプッシュダウンします。その後にGOTO文と同じように，行番号αの行にとんでいきます。RETURN文が来たらスタックをポップ・アップし，その中身に従ってもどっていきます。つまりG

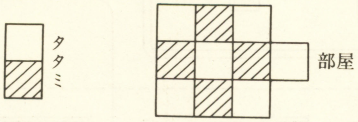
OSUBα と書かれた行の次へいきます。したがって，何回もGOSUBを実行するとスタックの中身はどんどんふくらんでいきます。最後には？　そうです。パンクする，というわけです。

なお，もちろんこのスタックはBASICの処理系が使っているシステムスタックであって，プログラムの中に書いたスタックとは違います。

休憩室 6　（まっすぐに進めない蟻）展開図で書きます。



休憩室 7　ヒントの問題をまず解いてみます。次の図のように部屋を市松模様にぬってみましょう。



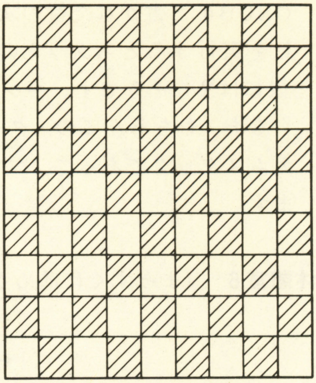
市松にぬった部屋

するとこの場合，□は6個，▨は4個となります。ところがタタミ1枚をおくと，その半分は□，半分は▨に必ずなってしまいます。したがって，□の数と▨の数が同じ部屋しかタタミをしけないということになります。この部屋は洋室にするしかないようですね。

将棋盤も市松にぬってみましょう。□が41個，▨が40個ですね。ところが1回桂馬とびをすると▨から□あるいは□から▨へ入れ換わるのです。ですから出発点が□なら，1回とぶと▨，2回とぶと□，……となり，偶数回目は□，奇数回目は▨になります。81回目

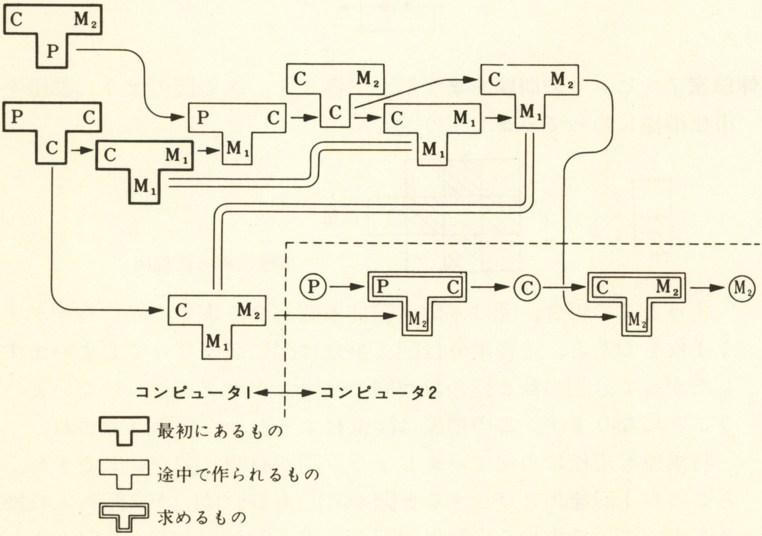
(＝9×9) に出発点にもどって来るのですから，そのときは奇数回目で▨です。ところが出発点は□だということで始めたのですから，これでは元にもどれませんね。

このように格子を市松模様にぬってパズルを解くのはエレガントで強力な方法です。市松模様のほかに縞模様などを使うと，うまくいくこともあります。しかし，計算機でこのような証明を発見するのは簡単ではありません。


市松にぬった将棋盤

(もちろんプログラムにこの事実を利用することはできます)。

コンパイラ・パズルの解答　図のように5回翻訳します。



模様を描く

洋服の模様になるような図形をパソコンに描かせてみます。基本になる長方形を規則的に敷きつめていきます。長方形はINPUT文で入力するようになっています。出力結果の一部を次に示します。

```
010 PRINT "<ﾓﾖｳ ｦ ｶｸ>"
020 DIM A$(400)
030 FOR T=1 TO 10
040  PRINT "MAZU, M*N KIHONZUKEI O IRERU"
050  PRINT "M,N"
060  INPUT M,N
070  PRINT N;"KO NO KIGO O IREYO"
080  FOR M1=1 TO M
090   INPUTS$
100   FOR N1=1 TO N
110    A$((M1-1)*N+N1)=MID$(S$,N1,1)
120    NEXT N1
130   NEXT M1
140  FOR R1=-1 TO 1 STEP 2
150   FOR R2=-1 TO 1 STEP 2
160    FOR S1=-1 TO 1 STEP 2
170     FOR S2=-1 TO 1 STEP 2
180      PRINT
190      PRINT R1;R2;S1;S2
200      FOR M9=1 TO 20/M
210       FOR M1=1 TO M
220        FOR N9=1 TO 40/N
230         FOR N1=1 TO N
240          N2=N1
250          IF R1^N9*S1^M9<0 THEN270
260          N2=N-N1+1
270          M2=M1
280          IF R2^N9*S2^M9<0 THEN 300
290          M2=M-M1+1
300          PRINT A$((M2-1)*N+N2);
310          NEXT N1
320         NEXT N9
330        PRINT
340        NEXT M1
350       NEXT M9
360      NEXT S2
370     NEXT S1
380    NEXT R2
390   NEXT R1
400  NEXT T
410 END
```

```
<ﾓﾖｳ ｦ ｶｸ>
MAZU, M*N KIHONZUKEI O IRERU
M,N
?5,10
   10 KO NO KIGO O IREYO
?    0 I   *
?  I=S=I  **
?  I X   ***
?   I I ****
?   I I*****

   -1    -1    -1    -1
*****I I     0 I   ******I I     0 I   *
**** I I   I=S=I  ****** I I   I=S=I  **
***   X I  I X   ******   X I  I X   ***
**  I=S=I   I I ******  I=S=I   I I ****
*   I 0     I I******   I 0     I I*****
   0 I   ******I I     0 I   ******I I
 I=S=I  ****** I I   I=S=I  ****** I I
 I X   ******   X I  I X   ******   X I
  I I ******  I=S=I   I I ******  I=S=I
  I I******   I 0     I I******   I 0
*****I I     0 I   ******I I     0 I   *
**** I I   I=S=I  ****** I I   I=S=I  **
***   X I  I X   ******   X I  I X   ***
**  I=S=I   I I ******  I=S=I   I I ****
*   I 0     I I******   I 0     I I*****
   0 I   ******I I     0 I   ******I I
 I=S=I  ****** I I   I=S=I  ****** I I
 I X   ******   X I  I X   ******   X I
  I I ******  I=S=I   I I ******  I=S=I
  I I******   I 0     I I******   I 0

   -1    -1    -1     1
*   I 0     I I******   I 0     I I*****
**  I=S=I   I I ******  I=S=I   I I ****
***   X I  I X   ******   X I  I X   ***
**** I I   I=S=I  ****** I I   I=S=I  **
*****I I     0 I   ******I I     0 I   *
   0 I   ******I I     0 I   ******I I
 I=S=I  ****** I I   I=S=I  ****** I I
 I X   ******   X I  I X   ******   X I
  I I ******  I=S=I   I I ******  I=S=I
  I I******   I 0     I I******   I 0
*   I 0     I I******   I 0     I I*****
**  I=S=I   I I ******  I=S=I   I I ****
***   X I  I X   ******   X I  I X   ***
**** I I   I=S=I  ****** I I   I=S=I  **
*****I I     0 I   ******I I     0 I   *
   0 I   ******I I     0 I   ******I I
 I=S=I  ****** I I   I=S=I  ****** I I
 I X   ******   X I  I X   ******   X I
  I I ******  I=S=I   I I ******  I=S=I
  I I******   I 0     I I******   I 0

   -1    -1     1    -1
  I I******   I 0     I I******   I 0
  I I ******  I=S=I   I I ******  I=S=I
 I X   ******   X I  I X   ******   X I
 I=S=I  ****** I I   I=S=I  ****** I I
   0 I   ******I I     0 I   ******I I
   0 I   ******I I     0 I   ******I I
 I=S=I  ****** I I   I=S=I  ****** I I
 I X   ******   X I  I X   ******   X I
  I I ******  I=S=I   I I ******  I=S=I
  I I******   I 0     I I******   I 0
  I I******   I 0     I I******   I 0
  I I ******  I=S=I   I I ******  I=S=I
 I X   ******   X I  I X   ******   X I
 I=S=I  ****** I I   I=S=I  ****** I I
   0 I   ******I I     0 I   ******I I
   0 I   ******I I     0 I   ******I I
 I=S=I  ****** I I   I=S=I  ****** I I
 I X   ******   X I  I X   ******   X I
  I I ******  I=S=I   I I ******  I=S=I
  I I******   I 0     I I******   I 0

   -1     1     1    -1
   0 I   **   I 0      0 I   **   I 0
 I=S=I  ****  I=S=I  I=S=I  ****  I=S=I
 I X   ******   X I  I X   ******   X I
  I I ******** I I    I I ******** I I
  I I**********I I    I I**********I I
  I I**********I I    I I**********I I
  I I ******** I I    I I ******** I I
 I X   ******   X I  I X   ******   X I
 I=S=I  ****  I=S=I  I=S=I  ****  I=S=I
   0 I   **   I 0      0 I   **   I 0
   0 I   **   I 0      0 I   **   I 0
 I=S=I  ****  I=S=I  I=S=I  ****  I=S=I
 I X   ******   X I  I X   ******   X I
  I I ******** I I    I I ******** I I
  I I**********I I    I I**********I I
  I I**********I I    I I**********I I
  I I ******** I I    I I ******** I I
 I X   ******   X I  I X   ******   X I
 I=S=I  ****  I=S=I  I=S=I  ****  I=S=I
   0 I   **   I 0      0 I   **   I 0

   -1     1     1     1
  I I**********I I    I I**********I I
  I I ******** I I    I I ******** I I
 I X   ******   X I  I X   ******   X I
 I=S=I  ****  I=S=I  I=S=I  ****  I=S=I
   0 I   **   I 0      0 I   **   I 0
  I I**********I I    I I**********I I
  I I ******** I I    I I ******** I I
 I X   ******   X I  I X   ******   X I
 I=S=I  ****  I=S=I  I=S=I  ****  I=S=I
   0 I   **   I 0      0 I   **   I 0
  I I**********I I    I I**********I I
  I I ******** I I    I I ******** I I
 I X   ******   X I  I X   ******   X I
 I=S=I  ****  I=S=I  I=S=I  ****  I=S=I
   0 I   **   I 0      0 I   **   I 0
  I I**********I I    I I**********I I
  I I ******** I I    I I ******** I I
 I X   ******   X I  I X   ******   X I
 I=S=I  ****  I=S=I  I=S=I  ****  I=S=I
   0 I   **   I 0      0 I   **   I 0

    1    -1    -1    -1
  I I*****   0 I   *  I I*****   0 I   *
  I I **** I=S=I  **  I I **** I=S=I  **
 I X   *** I X   *** I X   *** I X   ***
 I=S=I  **  I I **** I=S=I  **  I I ****
   0 I   *  I I*****   0 I   *  I I*****
*   I 0   *****I I  *   I 0   *****I I
**  I=S=I **** I I  **  I=S=I **** I I
***   X I ***   X I ***   X I ***   X I
**** I I  **  I=S=I **** I I  **  I=S=I
*****I I  *   I 0   *****I I  *   I 0
  I I*****   0 I   *  I I*****   0 I   *
  I I **** I=S=I  **  I I **** I=S=I  **
 I X   *** I X   *** I X   *** I X   ***
 I=S=I  **  I I **** I=S=I  **  I I ****
   0 I   *  I I*****   0 I   *  I I*****
*   I 0   *****I I  *   I 0   *****I I
**  I=S=I **** I I  **  I=S=I **** I I
***   X I ***   X I ***   X I ***   X I
**** I I  **  I=S=I **** I I  **  I=S=I
*****I I  *   I 0   *****I I  *   I 0

    1    -1    -1     1
   0 I   *  I I*****   0 I   *  I I*****
 I=S=I  **  I I **** I=S=I  **  I I ****
 I X   *** I X   *** I X   *** I X   ***
  I I **** I=S=I  **  I I **** I=S=I  **
  I I*****   0 I   *  I I*****   0 I   *
*   I 0   *****I I  *   I 0   *****I I
**  I=S=I **** I I  **  I=S=I **** I I
***   X I ***   X I ***   X I ***   X I
**** I I  **  I=S=I **** I I  **  I=S=I
```

参考になる本

この本の中身と関係があるのは，計算機（マイコン，パソコン）の本とパズル（ゲーム）の本です。計算機中心の本からパズル中心の本まで，参考になる本を順に並べてみましょう。

まず，最も大事なのは皆さんのパソコンのマニュアルです。これが基本になります。よく読んで機能を試してみましょう。手あかの付くまで読んでマスターすると，すごい力になります。

BASICの入門書は，たくさんあってきりがありませんので，ここではパズルに関係があるものを示します。

- 細井，嶋田：BASICによるプログラミング入門，産業図書
- 西村他：入門BASIC，オーム社

FORTRANのプログラムを扱ったパズルに関係ある本としては，

- 有澤誠：プログラミングレクリエーション，近代科学社
- 細井，嶋田：FORTRANによるプログラミング入門，産業図書
- 西村恕彦：人文科学のためのFORTRAN77，東京大学出版会

などがあります。

プログラムを書くときの「しつけ」については，

- カーニハン他（木村泉訳）：プログラム書法，共立出版

が参考になります。

ソフトウエアの設計法については，いろいろな考え方がありますが，ここではモジュール化を中心にしたものとして，

- Meyers（國友他訳）：ソフトウエアの複合／構造化設計，近代科学社

雑誌では，

- bit（共立出版）

があります。bitの「ナノピコ教室」という懸賞コーナーには，パズルを計算機で解かせる問題がよく出ます。また，計算機のソフトウエアを中心とした新鮮な話題が得られます。またマイコン雑誌では，RAM（廣

済堂出版）を初めとして，パズルをマイコンで解くという懸賞がいくつかあります。

パズルに関する本は，非常にたくさんありますが，ある程度複雑で考え甲斐のあるパズルが書かれているものを中心に紹介します。

● 別冊サイエンスⅠ～Ⅲ，日本経済新聞社
● 別冊数理科学，パズルⅠ～Ⅳ，サイエンス社
● ガードナー：数学ゲームⅠ，Ⅱ，講談社
● デュードニー：パズルの王様１～４，ダイヤモンド社
● 小城栄：パズルクイズル，光文社
● 芦ケ原伸之：パズル学新論，全国出版
● 資格試験研究会編：公務員試験教養分野別問題集「数的推理」，実務教育出版
● 志村光比古：新作パズルとゲーム，丸善出版サービスセンター

このほかに中公新書，KAPPA BOOKS（光文社），講談社ブルーバックスには多くのパズルの本があります。

パズルの同好会の会誌としては，

● 数芸パズル，数芸パズル愛好会（名古屋市東区石神堂町1-31-1）
● こんわかいニュース，パズル懇話会

があり，たくさんのパズルが発表されています。

〈著者紹介〉
小谷善行（こたに　よしゆき）
昭和46年12月東京大学工学部計数工学科数理工学コース卒業／昭和48年6月東京大学大学院修士課程修了（計数工学）／昭和52年3月東京大学大学院博士課程修了／同年4月東京農工大学工学部講師／昭和54年12月同大学工学部数理情報工学科助教授，現在に至る。
〈著書〉
入門BASIC＝共著（オーム社）その他

マイコンとパズルの世界
—BASICプログラミング頭の体操—

1981年11月25日　初版印刷
1981年12月10日　初版発行

定価980円

著　者　小谷善行
発行者　松山邦雄
発行所　産報出版株式会社

東京都台東区東上野 4-25-18（郵便番号 110）
電話 03（842）0511大代表・振替 東京1-58204

〈検印廃止〉

印刷＝株式会社廣済堂
製本＝秋元製本株式会社

© Printed in Japan，1981

万一乱丁・落丁がございましたら，書店または発行所でおとりかえいたします。

2034-531102（01）-2779

パソコン知識ABC

《楽しさとビジネス活用法》

矢矧晴一郎著

★B6判・216頁・定価980円

●パソコンをこれから使おうとしている初心者●パソコン導入直後のユーザー●パソコンを使いこなしているが他の人にやさしく説明する必要がある人に役立ちます。
【主な内容】パーソナルコンピュータの基本知識／ビジネスで使えるパソコン／早分かりハード・ソフトウエアの選び方／ビジネス利用の成功例・失敗例／他。

プログラマより愛をこめて！

マイコンゲーム21

●マイコン・キット遊ぶ本

岸田孝一編

★A5判・262頁・定価2200円

ＴＫ－80標準キット及びＴＫ－85ですぐ遊べるゲームプログラム集。また新しいゲームのプログラム作りや、拡張にもヒントが得られるゲーム集。

産報出版

マイコンとパズルの世界

BASICプログラミング頭の体操

小谷善行 著

定価980円

2034-531102-2779

マイコンとパズルの世界

小谷善行 著

産報出版

マイコンとパズルの世界

[illegible]

[illegible]